UNIVERSIDAD POLITÉCNICA DE VALENCIA

ESCUELA POLITÉCNICA SUPERIOR DE GANDIA

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓNES
ESPECIALIDAD SISTEMAS ELECTRÓNICOS

“ GRABADOR Y ENTRENADOR PARA PRÁCTICAS DE MICROCONTROLADORES CON PICs MEDIANTE USB “

TRABAJO FINAL DE CARRERA

Autor : Juan Carlos López Gordillo
Tutor : Tomás Carlos Sogorb Devesa

GANDIA 2013

INDICE

Objetivos y Plan de Trabajo

- Objetivos :

- Estudio y montaje de un programador y depurador para PICs compatible con el software de Microchip.
- Estudio y depuración de software y hardware utilizando el puerto USB.
- Realización de un entrenador de PICs usando el puerto USB.
- Adquirir un nivel de especialización mayor en la programación y manejo de los PICs.
- Estudio del modo de programación ICSP utilizado en los microcontroladores actuales.
- Realizar un sistema conjunto de Programador y Entrenador compatible con Microchip para ser utilizado en equipos actuales que utilicen Windows 7.

- Plan de trabajo :

- Estudio del Programador PicKit 2 original de Microchip, tanto a nivel Hardware como a nivel Software.
- Búsqueda de documentación.
- Diseño y realización del Hardware del Programador con algunas modificaciones.
- Comprobación del perfecto funcionamiento del Hardware del Programador realizado con el programa que Microchip suministra.
- Realización del Hardware del Nuevo Entrenador, cuidando la compatibilidad con el Entrenador anterior.
- Comprobación del perfecto funcionamiento del Hardware del Entrenador realizado con el programa que Microchip suministra.
- Realización de un Programa de ejemplo de aplicación. Control de un LCD utilizando el Entrenador y el Programador diseñado, usando el Software MPlab de Microchip.
- Realización de la Memoria del Proyecto.

PicKit 2 original de Microchip

Microchip Technology Inc. es una empresa líder de fabricación de microcontroladores, memorias y semiconductores analógicos que se encuentra situada en Chandler, Arizona, EE.UU.

Microchip Technology Inc. fabrica una familia de microcontroladores tipo RISC y derivados del PIC 1650. El nombre completo del actual PIC es en realidad, PICmicro, Peripheral Interface Controller (controlador de interfaz periférico).

El gran éxito obtenido por Microchip en el campo de los PICs ha sido sin duda la gran facilidad con la que cualquier persona puede acceder al software y hardware necesario para programar y/o utilizar los microcontroladores. Tanto el software como el hardware necesario para programar los PICs son de código abierto y se encuentran disponibles en la página web de Microchip.

Uno de los programadores que Microchip vende es el PicKit 2 el cual es posible adquirir con varios accesorios extra, el precio actual de sólo el programador es de 34.99 $ (web de Microchip 23/08/12), aunque también es posible adquirirlo a través de sus distribuidores en España (Farnell, RS-Amidata, etc…)

Microchip también suministra el esquema electrónico del programador y el software de programación totalmente gratuito y de código abierto disponible en su página web, así como un entorno completo de programación, depuración, compilación, etc… llamado MPLAB, el cual también se puede descargar de forma gratuita de la web de Microchip.

A continuación se indican los enlaces a fecha 23/08/12 con las últimas versiones aparecidas, hay que tener en cuenta que los enlaces pueden sufrir variaciones:

Web principal :

http://www.microchip.com/

Software MPLAB :

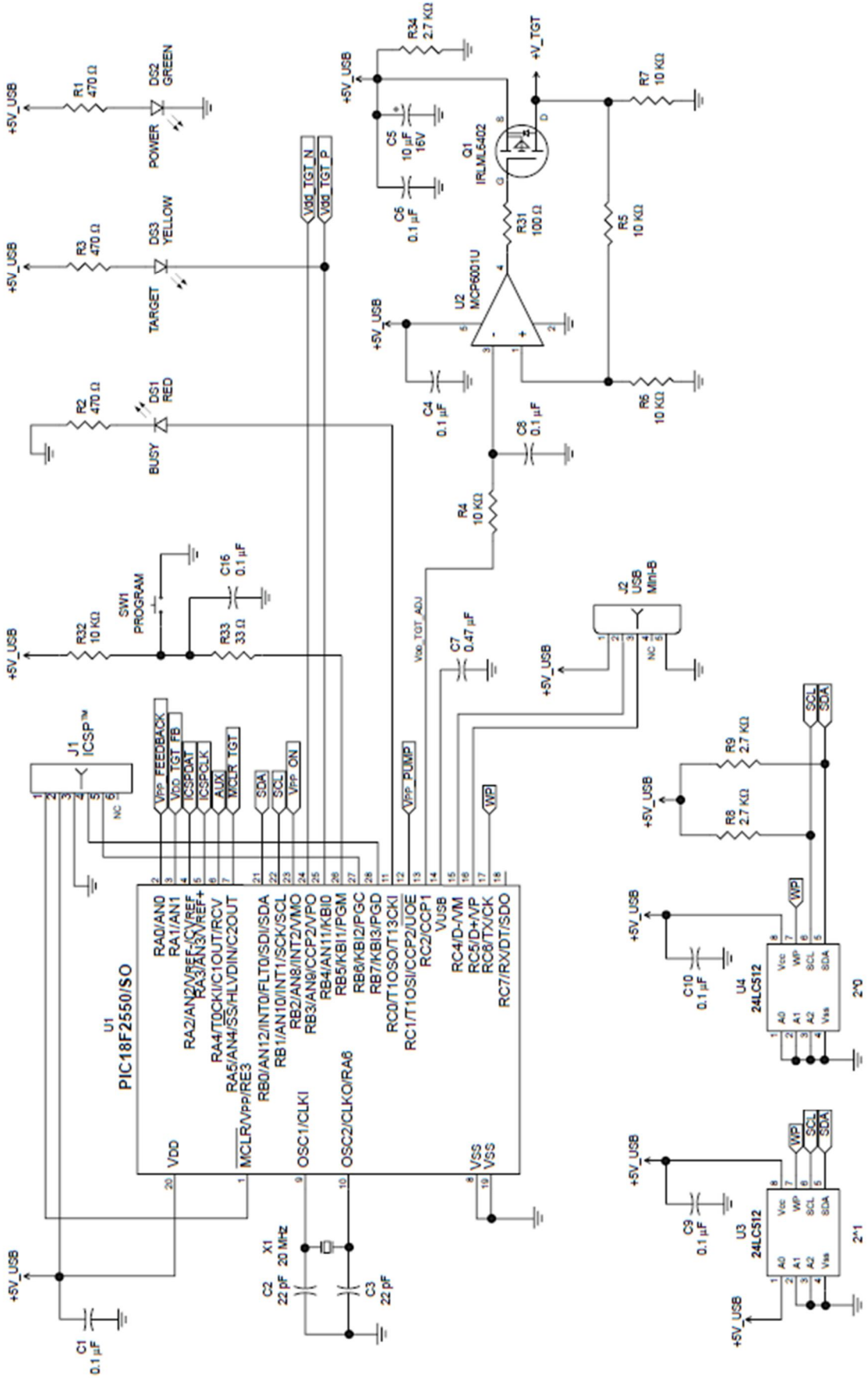
http://www.microchip.com/pagehandler/en-us/family/mplabx/

Programador PicKit 2 :

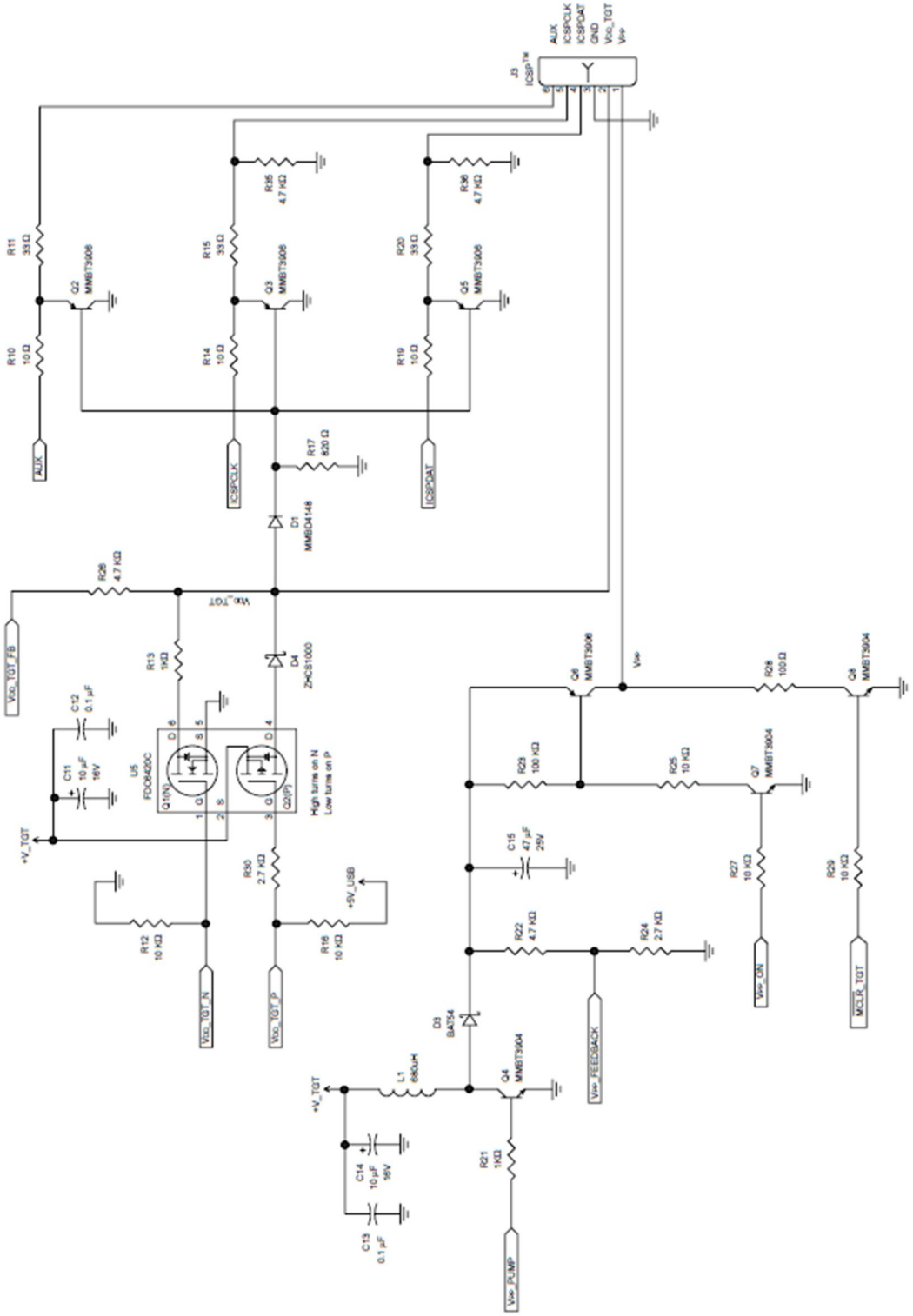
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en023805

En la página siguiente se expone el esquema electrónico original del Programador PicKit 2, el cual se ha tenido como referencia para la realización del programador USB del presente trabajo final de carrera. Se debe tener en cuenta que se ha intentado en todo momento que el programador diseñado sea un fiel reflejo del funcionamiento del PicKit 2 original de Microchip para poder utilizar el software y el firmware que Microchip proporciona.

Esquema del Programador PicKit 2 original de Microchip



Esquema del Programador PicKit 2 original de Microchip (continuación)



Podemos observar que a pesar de tener el esquema electrónico, tiene una alta densidad de componentes SMD y muchos de ellos son de difícil localización. Una de las prioridades que el Tutor del presente trabajo impuso es que fuera de fácil construcción con componentes fáciles de localizar, por lo tanto, no usaré componentes SMD, ya que son difíciles de soldar, e intentaré usar componentes tradicionales de fácil localización en tiendas de electrónica.

Viendo el esquema, podemos darnos cuenta que utiliza un PIC18F2550 para la programación del resto de PICs y para la comunicación con el Software del PC, por lo tanto en mi diseño usaré el mismo microcontrolador. También podemos observar que el diseño original dispone de 2 memorias 24LC512 que sirven para poder guardar el programa y prescindir del PC, en mi diseño las eliminaré, ya que es una característica no necesaria. La patilla 24 del PIC18F2550 es quien da la orden de empezar la grabación, por lo que toda la circuitería que se desprende de esta patilla, la mantendré, pero modificando los componentes por otros más tradicionales. La tensión +V_TGT es controlada mediante la patilla 13 del PIC, pero tras estudiar el esquema, he llegado a la conclusión que se puede dejar a +5V de forma constante, no interfiriendo en el funcionamiento del programador, aunque consuma más corriente, de esta manera nos ahorramos circuitería extra.

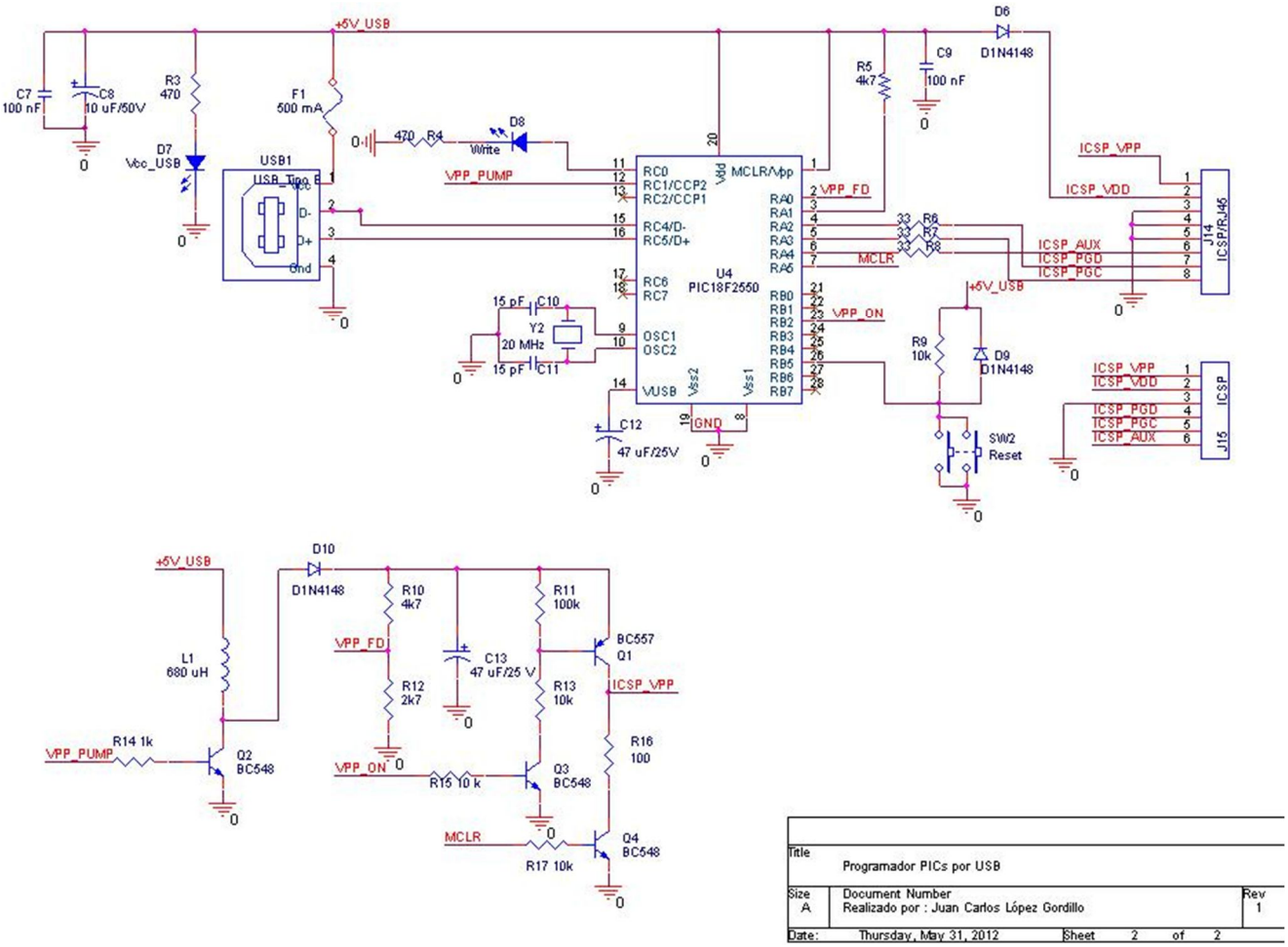
Por último, tras consultar la documentación de Microchip, se ha eliminado los transistores y la circuitería asociada a los pines 4, 5 y 6 del PIC porque el programador del presente trabajo se utilizará para programar el PIC 16F877 o similar y no se usará para memorias, por lo que dicha circuitería también es prescindible.

Con todo lo expuesto anteriormente, el circuito final que se va a utilizar en el trabajo final de carrera se encuentra basado en el esquema original del Pickit 2 de Microchip, pero eliminando la circuitería innecesaria para el proyecto, modificando y adaptando los componentes para utilizar otros más tradicionales.

Debemos fijarnos que al utilizar el mismo microcontrolador PIC18F2550 y tener las mismas conexiones básicas para realizar la programación ICSP (In Circuit Serial Programing), la conclusión final es, que el diseño realizado para el presente trabajo final de carrera, es totalmente compatible con el PicKit 2 original de Microchip en la programación de PICs se refiere, por lo tanto utilizará el mismo firmware que el PicKit 2 original, pudiéndose utilizar el mismo software y va a ser identificado exactamente igual que si tuviéramos el PicKit 2 original.

En la página siguiente se muestra el esquema resultante del programador usado en el presente trabajo final de carrera.

Grabador USB compatible con PicKit 2 de Microchip



Como se observa en el esquema, el diseño se basa en el microcontrolador PIC16F2550, igual que el PicKit 2 original de Microchip, al que se le ha quitado la circuitería innecesaria, se le ha modificado los componentes para que sean más accesibles y se han añadido algunas mejoras que se han considerado.

Las mejoras que se han añadido son las siguientes :

- Añadido un fusible rearmable de 500 mA en la alimentación del puerto USB como medio de protección.

- Añadido el diodo D6 en la patilla 2 de alimentación al entrenador para evitar daños provocados por alimentaciones externas al programador.

- Añadido un conector RJ45 para poder conectarse con el entrenador a través de un cable directo de Ethernet.

Funcionamiento del programador USB compatible con PicKit 2 de Microchip :

El PIC18F2550 se encuentra programado previamente con un Firmware versión 2.32 proporcionado por Microchip de su página web :

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en023805

El código de este Firmware lo expondré más adelante, ahora únicamente explicaré el funcionamiento del hardware del programador diseñado.

El PIC18F2550 se encuentra correctamente alimentado a través de las patillas 20 (Vdd), las patillas 8 y 19 (GND)y la patilla 1 (MCLR) que se encuentra a +Vcc (nivel alto), igualmente dispone de un cristal de cuarzo de 20 MHz junto con sus condensadores de 15 pF correspondientes para realizar un oscilador que funcionará como reloj conectado en las patillas 9 y 10 del PIC. Además al ser un microcontrolador que ya tiene implementado el hardware necesario para la comunicación USB, se ha conectado un conector USB en las patillas 15 (D-) y 16 (D+) para realizar la comunicación con el PC mediante el puerto USB, que según el datasheet del PIC es necesario conectar un condensador electrolítico de 47 µF en la patilla 14 (VUSB) del PIC.

Para mantener la compatibilidad con el firmware que Microchip suministra se observa que utiliza la patilla 26 (RB5) del PIC para provocar el Reset en el programa, por lo tanto en dicha patilla, se ha diseñado un circuito de Reset con una resistencia (R9) pull-up de modo que cuando se pulse SW2, se introduce un “0” (nivel bajo) y esto provocará el Reset del Firmware, si no se pulsa SW2, en la patilla 26 se tendrá un “1” (nivel alto), lo cual no hará ningún efecto. Observamos también que el firmware utiliza la patilla 11 (RC0) para indicar que se encuentra ocupado “Busy”, lo cual significa que se encuentra programando, por lo tanto conectado a esa patilla se encuentra un led con su resistencia limitadora para indicar dicha función.

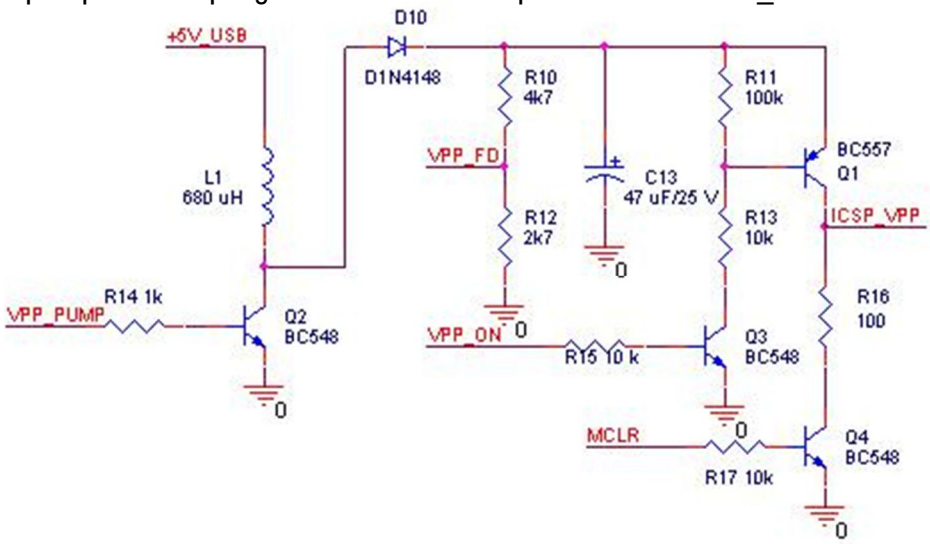
La patilla 3 (RA1) se ha comprobado que debe estar a “1” (tensión alta) para que realice la función de programación, por lo tanto se lleva a +Vcc a través de R5. La patilla 2 (RA0) recibe una señal continua entre 0V y $V_{C_{13}} \cdot \frac{R_{12}}{R_{12}+R_{10}} = V_{C_{13}} \cdot \frac{2.7}{2.7+4.7} = V_{C_{13}} \cdot 0{,}36487$ que proviene del divisor de tensión compuesto por R10 y R12, en función del circuito formado por el transistor Q2, la bobina L1, el diodo D10, el condensador C13 y controlado todo por la patilla

12 (CCP2) del PIC. Este circuito está diseñado para que el PIC sepa en todo momento si se encuentra en modo Programación o en modo ejecución.

Las patillas 4 y 5 (RA2 y RA3) son las encargadas de enviar con el protocolo adecuado el programa al PIC que queramos programar.

La patilla 6 (RA4) es una salida auxiliar para controlar mediante software la alimentación del PIC que queramos programar (en nuestro entrenador no se utilizará, pero se ha dejado esta opción para que sea compatible con PicKit 2 original de Microchip.

Las patillas 2 “Vpp_FD” (RA0), 7 “MCLR” (RA5), 12 “Vpp_PUMP” (CCP2) y 23 “Vpp_ON” (RB2) controlan a un circuito convertidor cc-cc elevador (Boost) que se encarga de poner en modo ejecución, modo programación o resetear al PIC que queramos programar mediante el pin de salida “ICSP_VPP”.



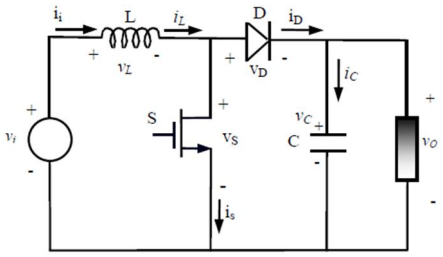
Como podemos observar este circuito se encuentra alimentado por +5V suministrados por el puerto USB. El PIC a través de su patilla 12 (CCP2) “Vpp_Pump” envía una señal cuadrada modulada por anchura de pulsos (PWM) a la base del transistor Q2, el cual actúa de conmutador de señal de la bobina L1. El diodo D10 actúa como rectificador de señal, cargando el condensador C13 con una tensión continua comprendida entre 0V y 12V en función de la señal PWM enviada por el PIC. Una parte de esa señal (aprox. 40 %) es leído por la patilla 2 (RA0) “Vpp_FD” del PIC que en función de la tensión leída hará que el PIC se comporte de una forma u otra, enviando al Software del PC información. Si enviamos la orden de Programación desde el PC, el PIC activará la señal “Vpp_Pump” y si recibe una tensión suficiente en la señal “Vpp_FD”, activará la señal “Vpp_ON” (patilla 23, RB2), la cual hará pasar a saturación al transistor Q3, y este a su vez hará lo mismo con Q1, el cual hará que en ICSP_VPP tengamos aproximadamente 12 V (tensión de programación). Cuando termine de programar al PIC conectado en ICSP, el PIC del programador modulará la señal “Vpp_PUMP” para tener en “ICSP_VPP” una tensión de +5V. En cualquier momento y

usando la señal "MCLR" conectada en la patilla 7 (RA5), el PIC puede Resetear al PIC conectado en ICSP.

Análisis de un Convertidor cc-cc (Boost)

Este tipo de convertidor, denominado boost o step-up en la topología anglosajona, es utilizado cuando deseamos un aumento de la tensión de salida con relación a la tensión de entrada. La polaridad de la tensión de salida es la misma que la de entrada. El ruido generado en la salida es alto debido a los pulsos de corriente suministrados al condensador de salida C. El ruido generado a la entrada es bajo porque la inductancia L, directamente conectada a la tensión de entrada, mantiene la variación de corriente de entrada sin pulsos.

Análisis en régimen permanente y modo de conducción continua



En la figura se muestra la topología del convertidor elevador (boost) donde se ha dibujado la fuente de tensión de entrada sin especificar su forma. Se considerará en todos los análisis que la tensión de entrada v_i es una tensión unipolar con un cierto rizado.

El convertidor tiene dos modos de funcionamiento :

- modo de conducción continua
- modo de conducción discontinua

Para simplificar, analizaremos el modo de conducción continua. En este modo de funcionamiento, la corriente por la bobina del convertidor nunca se anula, de manera que siempre conduce alguno de los interruptores de potencia del convertidor, es decir, o conduce el transistor o conduce el diodo. Nótese que ambos a la vez no pueden conducir dado que el diodo ve una tensión ánodo-cátodo negativa cuando conduce el transistor.

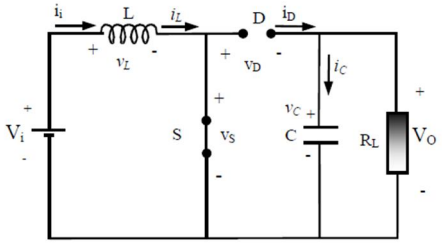
Para simplificar y facilitar el entendimiento, vamos a suponer estado estable, o sea, la tensión en el condensador es constante y la corriente de salida también. El transistor se hace funcionar en la región óhmica y en corte a una frecuencia f_S, de modo que, o bien el transistor conduce, o bien el diodo conduce.

Cuando el transistor conduce, la inductancia está directamente conectada a la tensión de entrada y suponiendo que la tensión de salida sea mayor que la tensión de entrada (un hecho real en el convertidor BOOST), el diodo estará inversamente polarizado y la tensión en el condensador suministrará una corriente a la carga (resistencia).

Cuando se corta el transistor, el diodo conduce y el inductor suministra corriente a la salida (resistencia y condensador). Esa corriente debe ser tal que reponga las cargas perdidas por el condensador en el instante anterior y suministre la corriente a la resistencia.

Se indicará a partir de ahora que un interruptor está conduciendo substituyéndolo simplemente por un hilo, mientras que su estado de bloqueo se indicará por un circuito abierto.

El circuito equivalente del convertidor cuando el transistor está conduciendo se muestra en la siguiente figura :



Se supone las siguientes condiciones iniciales :

$V_i = v_i = V_o = v_o \approx$ constante
$V_{Son} = V_{Don} \approx 0$
$T_S =$ constante

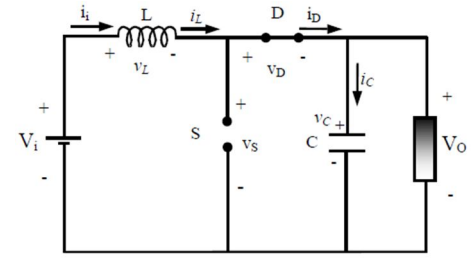
Se consideran despreciables las variaciones de las tensiones de entrada y de salida, así como las caídas de tensión en el transistor y el diodo cuando conducen. Además, se considera en este análisis que el circuito de control mantiene constante el período de conmutación.

$V_i = L \cdot \frac{d_{iL}}{d_t}$, integrando en el intervalo t ε [0, T_{ON}]

$$i_L(t) - i_L(0) = \frac{V_i \cdot t}{L}$$

Esta expresión nos indica una evolución constante de la corriente.

El circuito equivalente del convertidor cuando el transistor está conduciendo se muestra en la siguiente figura :



Si seguimos el mismo planteamiento que para el caso anterior, llegamos a la expresión siguiente :

$$V_i - V_o = L \cdot \frac{d_{iL}}{d_t}$$

Si se integra en el intervalo t ε [T_{ON}, T_S] queda la siguiente expresión :

$$i_L(t) - i_L(T_{on}) = \frac{(V_i - V_o) \cdot (t - T_{on}}{L}$$

La evolución de la corriente en la inductancia vuelve a ser lineal. A diferencia del caso anterior (intervalo T_{on}), la pendiente de i_L es ahora negativa, lo cual responde al fenómeno físico de que la energía en la inductancia está disminuyendo. Por tanto, durante el intervalo T_{on} la inductancia recibe energía de la fuente de entrada, pues i_L es una función creciente, mientras que durante T_{off} la inductancia entrega energía a la carga, al ser i_L una función decreciente.

Función de transferencia

En régimen permanente se ha de cumplir que el incremento de la corriente de la bobina durante T_{on} debe ser igual al decremento de la corriente de la bobina durante T_{off}, dado que:

$$i_L(0) = i_L(T_S)$$

Se deduce entonces la igualdad del incremento y decremento de la corriente en la inductancia durante T_{on} y T_{off}, respectivamente:

$$i_L(T_{on}) - i_L(0) = -\,(i_L(T_s) - i_L(T_{on}))$$

Tenemos entonces las siguientes expresiones ;

$$i_L(t) - i_L(0) = \frac{V_i \cdot t}{L} \qquad \text{para } t \in [\,0, T_{ON}\,]$$

$$i_L(t) - i_L(T_{on}) = \frac{(V_i - V_o)\cdot(t - T_{on}}{L} \qquad \text{para } t \in [\,T_{ON}, T_S\,]$$

Por lo tanto ;

$$\frac{V_i \cdot T_{on}}{L} = -\left[\frac{(V_i - V_o)\cdot(T_S - T_{on})}{L}\right]$$

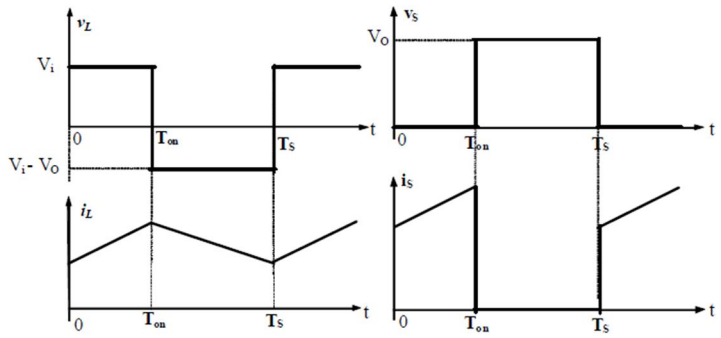
Resultando la función de transferencia siguiente :

$$\frac{V_o}{V_i} = \frac{1}{1-D}$$

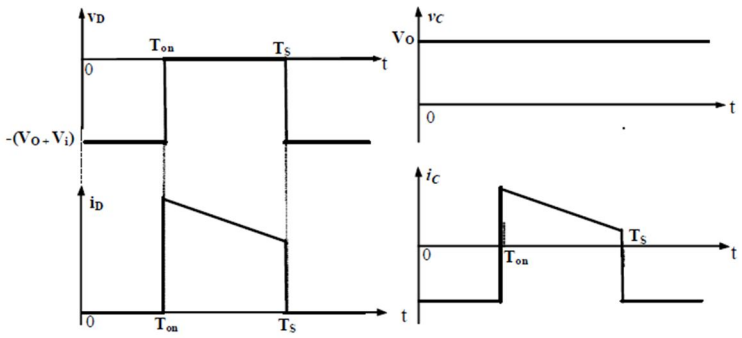
siendo $D = \frac{T_{on}}{T_s}$ es decir, $D \in [\,0, 1\,]$

Debemos fijarnos que la tensión de salida es siempre mayor o igual a la entrada, ya que la relación de conducción $D \in [\,0, 1\,]$.

Las formas de onda son las representadas en las siguientes figuras :



Hay que notar la diferencia entre la forma de onda de la corriente por el interruptor y la de la corriente por la inductancia. La corriente por el interruptor es pulsante, en el sentido de que, dentro de cada período, hay un intervalo en que es nulo y otro en que no es nulo (y normalmente elevado).



Como se ha visto un convertidor elevador (boost) puede subir la tensión de salida sin necesidad de un transformador. Debido a que sólo tiene un transistor, su eficiencia es alta. La corriente de entrada es continua. Sin embargo, a través del transistor de potencia debe fluir una corriente de pico elevada. La tensión de salida es muy sensible a cambios en la relación de conducción D (duty ratio) y puede resultar difícil estabilizar el regulador.

Entrenador para el PIC 16F877

Las condiciones expuestas por el Tutor para el diseño del entrenador es que fuera compatible con los módulos diseñados para el entrenador anterior, que se pueda programar usando un PicKit 2 original o el diseñado para el presente trabajo y que tenga la posibilidad de elegir varias entradas de alimentación, en función del consumo que se necesite.

Con dichas condiciones, se ha diseñado un entrenador con un conector compatible con el grabador PicKit 2 y el diseñado para este trabajo, además debido a que en el grabador diseñado se ha añadido un conector RJ45, también se ha añadido este conector para la conexión mediante un cable de red Ethernet directo.

Las entradas de alimentación se ha decidido tener 3 fuentes que pueden ser utilizadas de forma independiente, pero siempre se debe tener en cuenta que en el proceso de grabación el PIC16F877 del entrenador, será alimentado siempre a través del grabador.

Entradas de alimentación diseñadas en el entrenador :

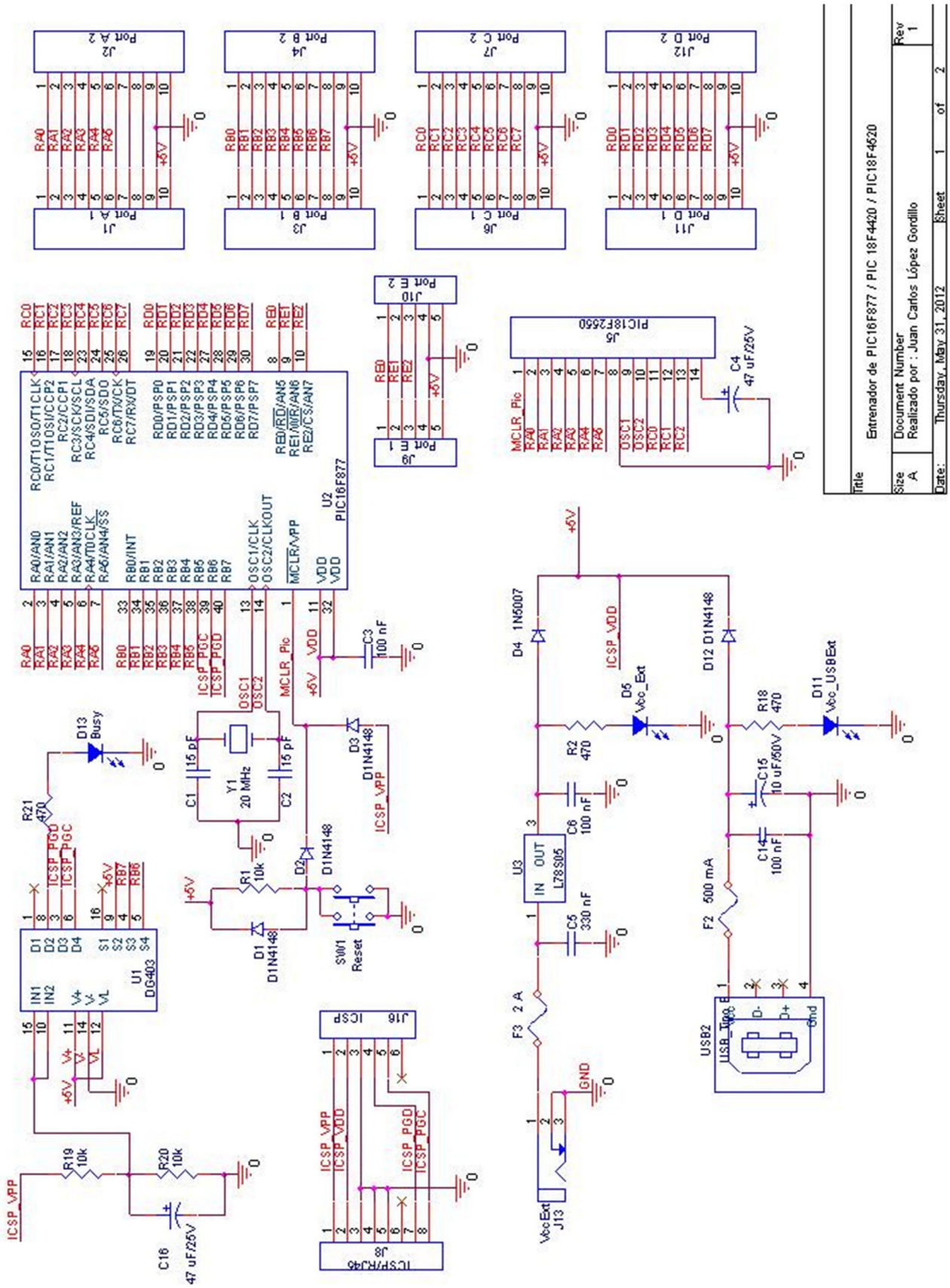
- Entrada de 5 V a 16 V y 2 A como máximo, compatible con las Fuentes de alimentación utilizadas en el entrenador anterior.
- Entrada de 5V y 500 mA como máximo procedente de la conexión USB de un PC o de cualquier otra fuente que disponga de conexión USB.
- Entrada de alimentación de 5V y 500 mA como máximo procedente del grabador PicKit 2 original o el diseñado.

Todas las entradas de alimentación se encuentran protegidas con fusibles rearmables.

Además se han añadido diodos leds para indicar cual de las entradas de alimentación se encuentra utilizada y un conmutador electrónico que aíslan las patillas que se utilizan en la grabación ICSP del PIC del conector externo cuando se transfiere el programa desde el programador al entrenador y conectan dichas patillas al conector externo cuando se ejecute el programa, de esa manera disponemos de todas las patillas del PIC para ser utilizadas por los módulos que se usen en las prácticas. Un led rojo nos indicará que se está transfiriendo el programa y dichas patillas están conectadas al programador y desconectadas del conector exterior.

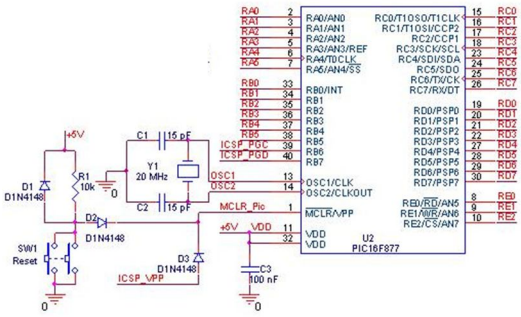
En la hoja siguiente se muestra el esquema que se ha diseñado.

Esquema del entrenador del PIC16F877

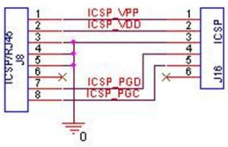


Como se puede apreciar, el esquema se compone del zócalo para el PIC16F877 o cualquier PIC de 40 patillas que sea compatible, un zócalo para el PIC18F2550 o cualquier PIC de 28 patillas compatible, los conectores compatibles para conectar los módulos que el Profesorado dispone del entrenador anterior, el conmutador electrónico DG403 para poder utilizar las patillas RB6 y RB7 del PIC en los conectores de salida, las cuales se utilizan para programar el PIC por ICSP y las entradas de la fuentes de alimentación externas.

Funcionamiento del Entrenador :

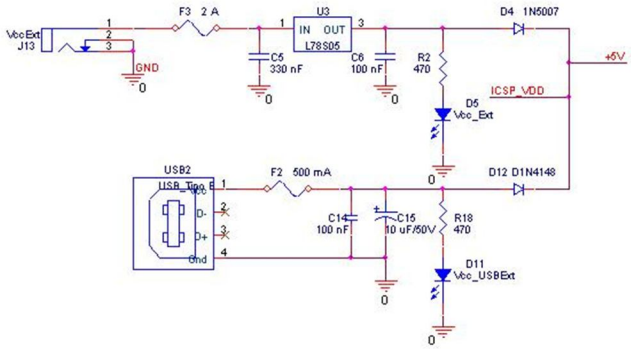


El zócalo principal, donde ira ubicado el PIC16F877, tiene en las patillas OSC1 (13) y OSC2 (14) conectado el cristal de cuarzo de 20 MHz y 2 condensadores de 15 pF para ser utilizado como reloj del sistema, además se puede observar que en las alimentaciones (patillas 11 y 32) se encuentra conectado con condensador de 100 nF para estabilizar la señal continua de alimentación. Debemos fijarnos también que todos los pines de los Puertos A, B, C, D, E se encuentran conectados a los conectores externos para ser conectados a los módulos correspondientes que los Alumnos utilizarán para realizar las Prácticas, a excepción de los pines 39 (RB6) y 40 (RB7), los cuales son utilizados por el programador para la grabación ICSP. La patilla 1 (MCLR/VPP) del PIC es utilizada para poner al PIC en modo programación (12V), resetearlo (0 V) o alimentarlo (5 V), por lo tanto a esa patilla se le ha conectado el circuito con el Switch 1 (Reset), el diodo D1 y la resistencia pull-up R1 que si en ICSP_VPP no tenemos ninguna diferencia de potencial, hará que el PIC reciba 0V en la patilla 1, reseteando al PIC, pero si a través del diodo D3 se dispone de una tensión mayor de 0 V, no producirá ningún efecto.

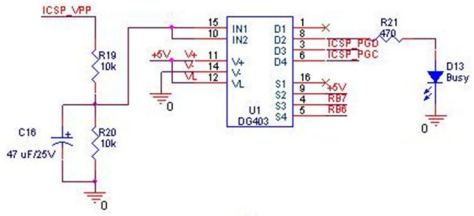


Esta tensión puede ser modificada por el programador, por lo que si se encuentra en modo programación, tendremos 12 V, si se encuentra en modo ejecución, tendremos 5 V y si el programador no se encuentra conectado, se encontrará conectado a 5 V a través de la resistencia pull-up, pudiendo resetear al PIC con el pulsador de Reset antes descrito.

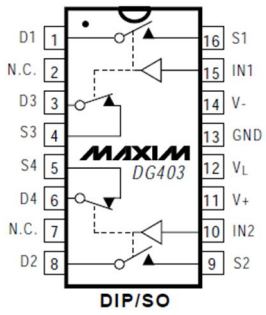
La alimentación, la recibe a través del mismo conector ICSP que proviene del programador ICSP_VDD (si está conectado) o a través del siguiente circuito :



Podemos observar que las etiquetas ICSP_VDD y +5V se encuentran unidas, por lo que si el Programador se encuentra conectado al entrenador, es el propio Programador quien alimenta al entrenador, pero también disponemos de la posibilidad de alimentar al entrenador a través de una fuente externa VccExt, la cual se ha añadido para poder utilizar la Fuente de alimentación que disponía el entrenador anterior (16V/2 A) y así poder conectar cargas de gran consumo de corriente, ya que se ha previsto un regulador L78S05, un diodo D4 1N5007 y un Fusible rearmable F3 para poder suministrar una corriente de 2 A, también se ha añadido la posibilidad de ser alimentado externamente por un dispositivo de 5V c.c. / 500 mA a través del conector USB2 (por ejemplo un puerto USB de un PC). Nos debemos fijar también que se dispone de unos leds indicadores.

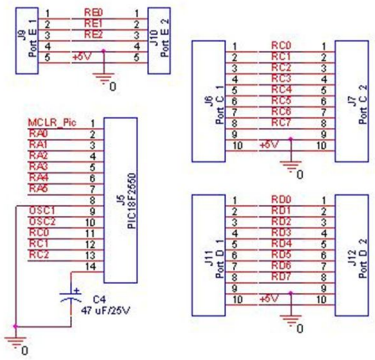


El circuito formado por el integrado DG403, realiza la función de detectar cuando el programador pone la línea ICSP_VPP a (12V), por lo que en las patillas 15 (IN1) y 10 (IN2) del DG403, tendremos 6V (debido al divisor de tensión R19 y R20), y esto hará que las patillas RB6 y RB7 del PIC que nos encontramos programado se encuentren aisladas del conector del Puerto B, además se el led D13 lucirá indicando que se encuentra ocupado (Busy).



Aquí podemos ver la constitución interna del DG403, donde podemos observar que las patillas 15 y 10 son las que controlan la conmutación, cuando nos encontramos en modo programación estas patillas recibirán tensión alta, la cual hará que los interruptores de las patillas (3,4) y (5,6) se abran por lo que las conexiones de los conectores RB6 (patilla 5) y RB7 (patilla 4) se encuentran abiertas. Sin embargo en modo ejecución, los interruptores se encuentran cerrados, por lo que los pines del conector del puerto B, se encuentra conectado al PIC, pudiéndose utilizar sin ningún problema en las Prácticas que los Alumnos se encuentren realizando.

Por último podemos observar aquí, los conectores de los Puertos del PIC, los cuales en todas las conexiones se dispone de los bits de datos y de alimentación (Vcc y Gnd), fijémonos también en el Puerto E que es de menor número de Pines pero se ha mantenido el mismo criterio.



Además se observa también el zócalo para el PIC18F2550, este zócalo comparte con el zócalo principal del PIC16F877 las patillas 26 a la 40, y las patillas de 1 a la 14 del zócalo del PIC18F2550 se han conectado a las patillas correspondientes del zócalo principal para poder utilizar un solo reloj y poder compartir el resto de pines.

El objetivo de añadir este zócalo que se ha colocado dentro del zócalo principal con el propósito de que no se puedan conectar a la vez el PIC16F877 y el PIC18F2550 es que si el usuario quiere construirse otro Programador, podrá utilizar este zócalo para programar al PIC18F2550 con el firmware adecuado e insertar de forma rápida el nuevo PIC en el zócalo del nuevo Programador construido. Además, también se podría utilizar para utilizar el Entrenador con otro PIC compatible que no sea necesariamente el PIC16F877, por ejemplo otro PIC18F2550.

Para concluir la explicación del Entrenador, debo decir que debido a que en el Entrenador anterior se disponía de una tira de pines y una tira de conectores para cada Puerto conectados en paralelo. Esta configuración se ha mantenido como compatibilidad con el Entrenador anterior.

Diseño de las Placas de Circuito Impreso

Una exigencia que el Tutor me comentó fue que, el alumno pueda fabricarse de forma sencilla el Programador y el Entrenador para así poder si lo desea construírselo el mismo y llevárselo a su casa para poder utilizarlo y realizar Prácticas en su hogar.

Por tal motivo, el diseño del PCB del Programador y del Entrenador se ha realizado con componentes fáciles de adquirir y en montaje DIP, ya que la tecnología SMD necesita de herramientas más sofisticadas y dichas herramientas y los componentes son más difíciles de adquirir por el Alumnado.

Además se ha pensado en la posibilidad de que se puedan realizar el Programador y el Entrenador por separado o conjuntamente, debido a que cabe la posibilidad de que se quiera únicamente construir el Programador, únicamente el Entrenador o el sistema completo.

Por ejemplo, en un principio, para el taller o para un Alumno puede ser interesante construirse el sistema completo y tenerlo como una única placa, ya que el propósito es realizar una serie de Prácticas en la que Programador y Entrenador van a ser utilizados de forma conjunta.

Sin embargo, cabe la posibilidad que por ejemplo, en el taller o el propio Alumno disponga de un PicKit 2 comprado de Microchip, por lo que en ese caso sólo sería necesaria la construcción del Entrenador.

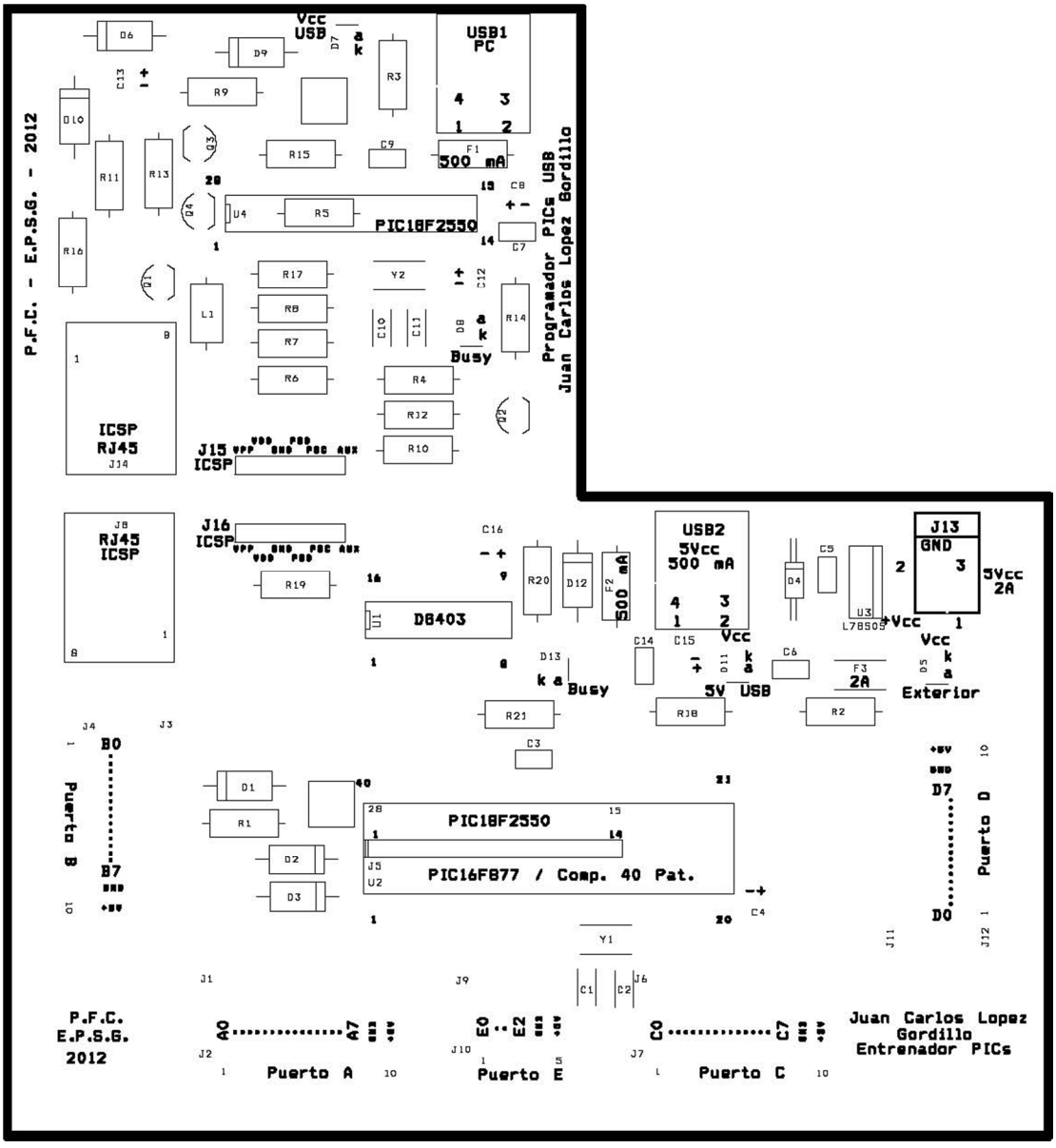
También podría suceder que se necesite algún Programador adicional, bien porque se ha averiado otro Programador o bien porque el uso que se le va a dar no es para ser utilizado con el Entrenador, en ese caso, sólo sería necesaria la construcción del Programador.

Por los motivos anteriormente citados, se ha realizado el diseño para que se pueda separar de forma sencilla Programador y Entrenador, simplemente cortando la PCB en la unión de ambos circuitos.

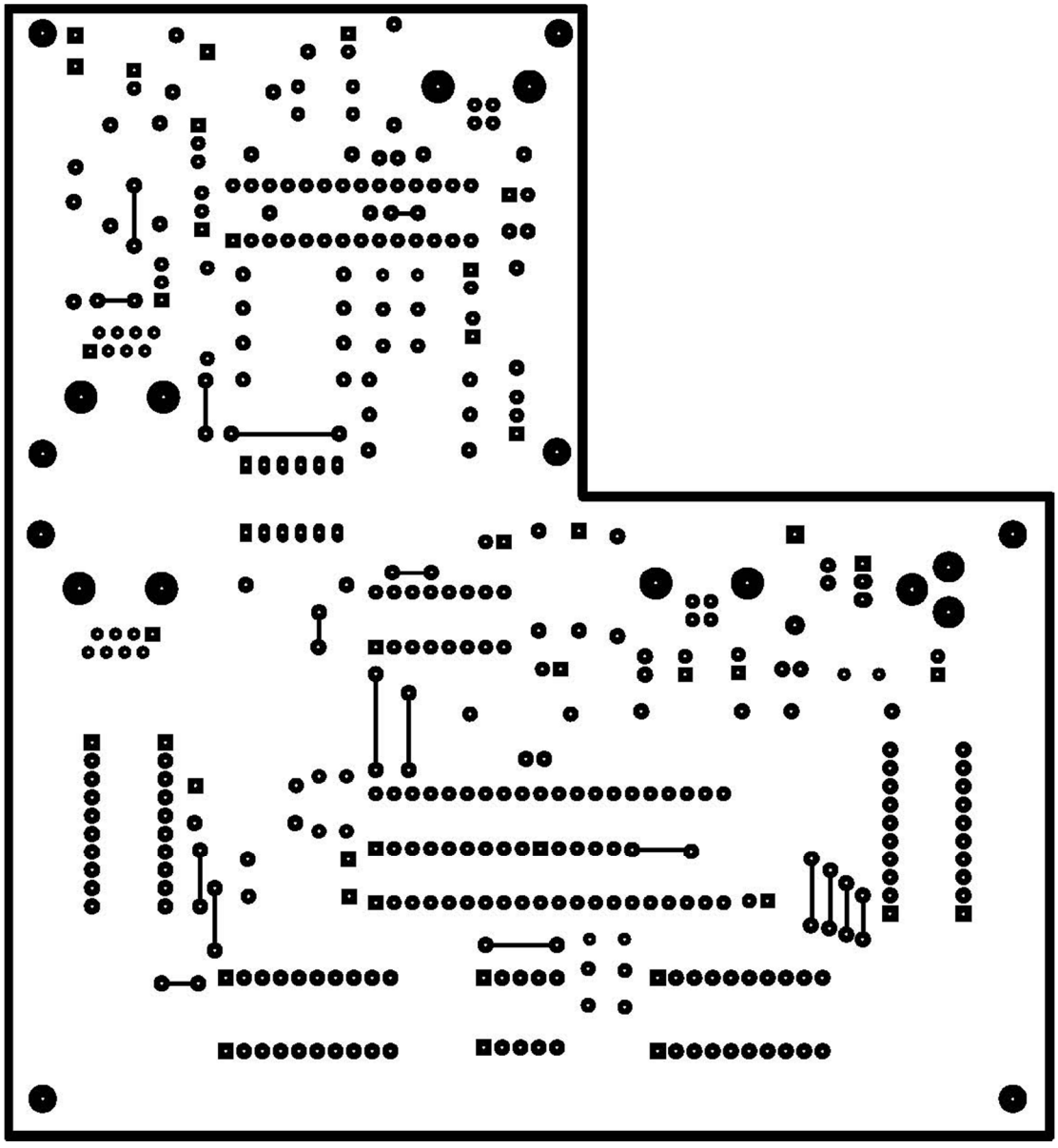
Cabe mencionar que en caso de querer tener los 2 circuitos unidos en una sola PCB o si no se quiere utilizar la conexión mediante cable Ethernet directo, se puede prescindir del conector RJ45 del Programador y del Entrenador.

Placa de Circuito Impreso del Programador y del Entrenador de Pics por USB :

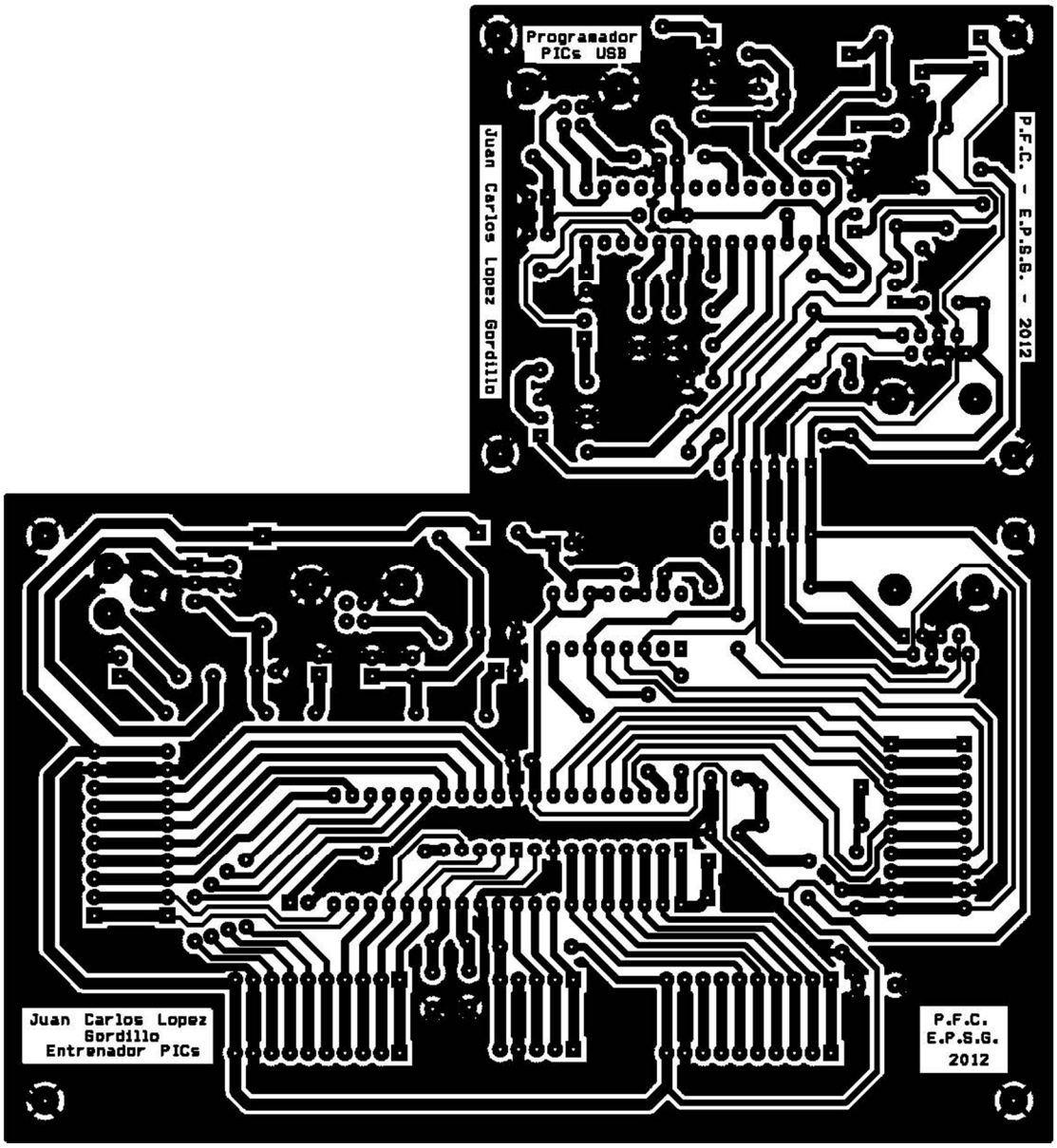
- Cara de componentes (sólo componentes) :



- Cara de componentes (puentes a realizar) :

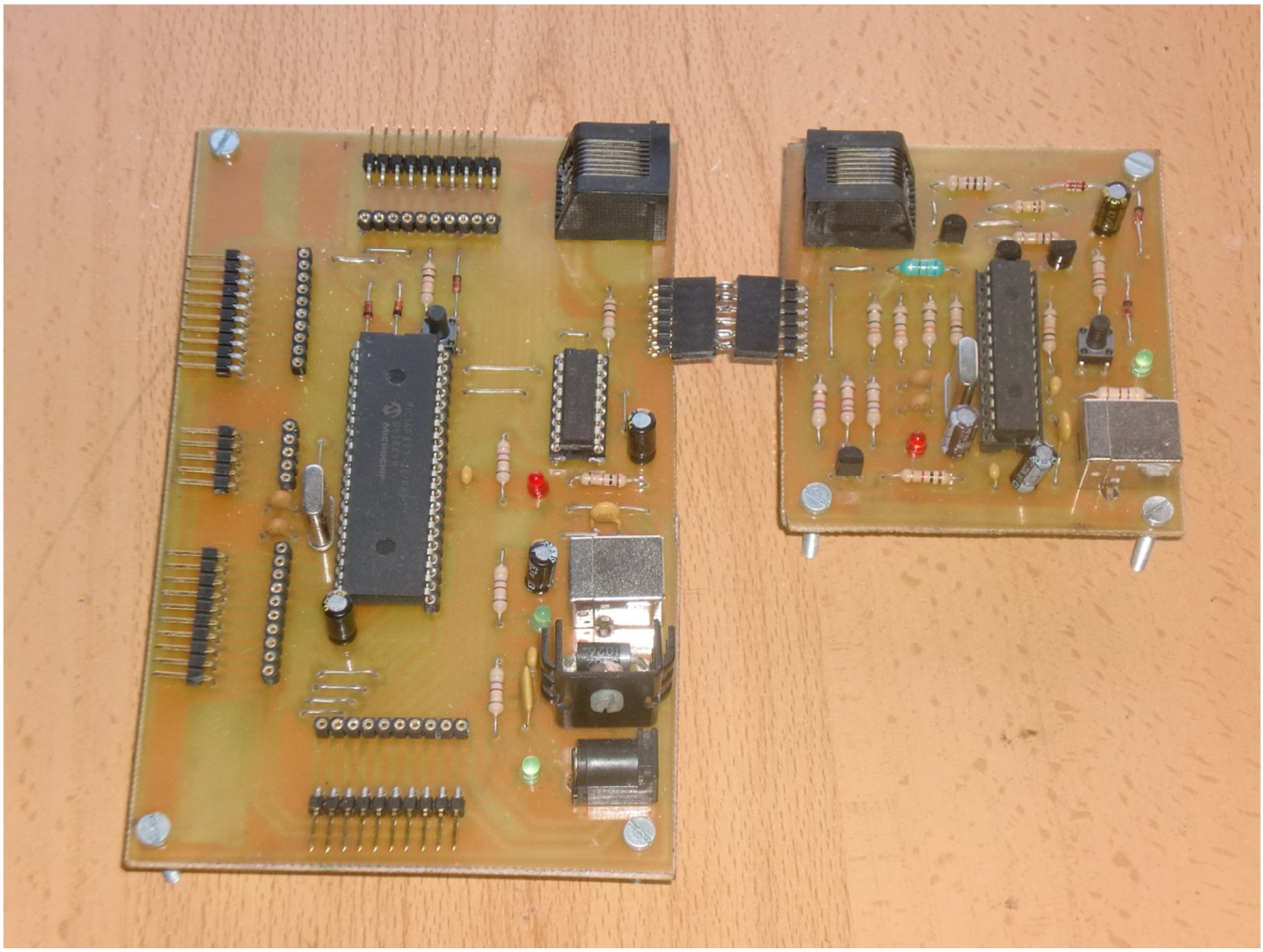


- Cara de Pistas :

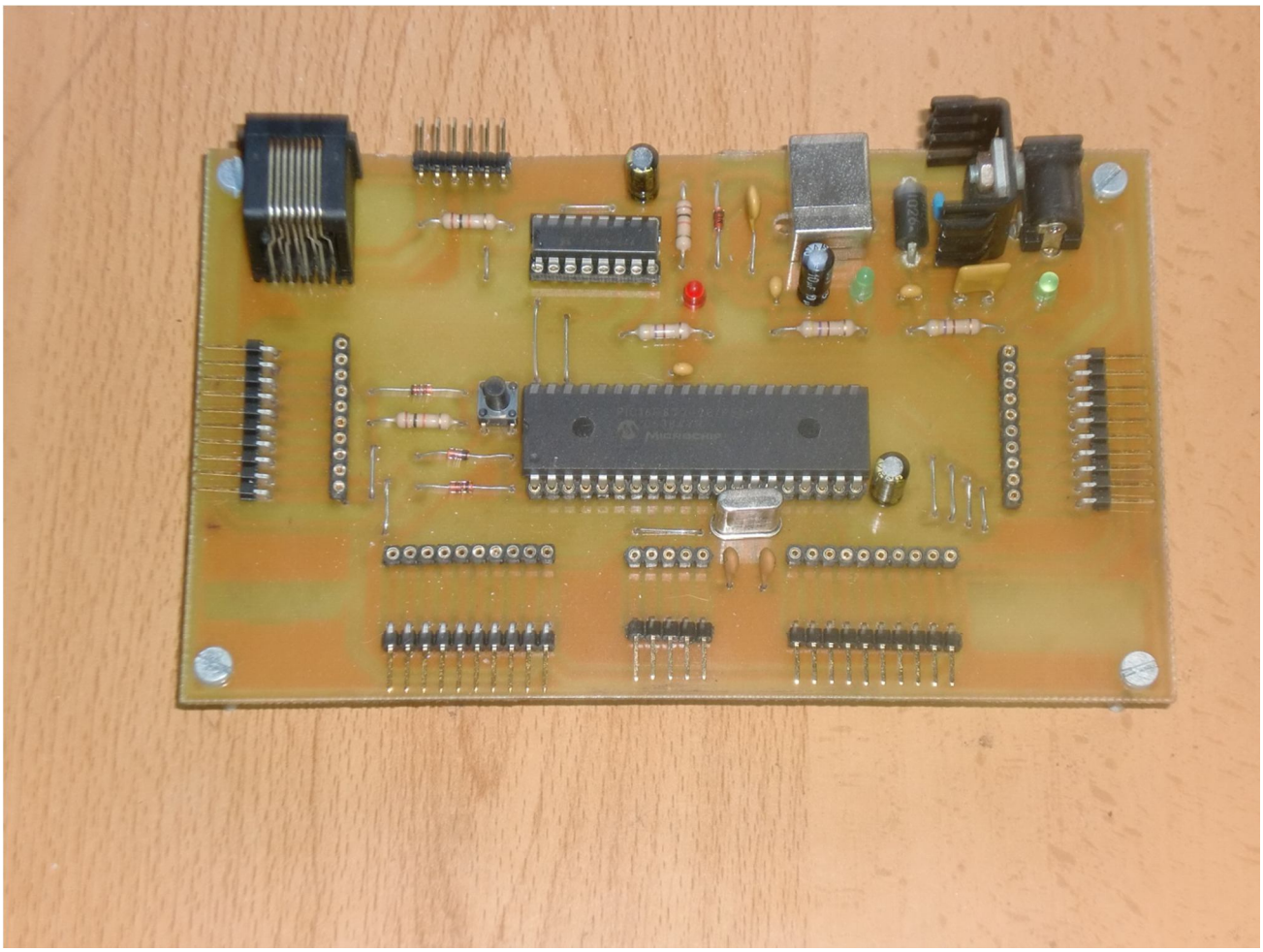


A continuación se muestran algunas fotografías tomadas del programador y del entrenador realizado :

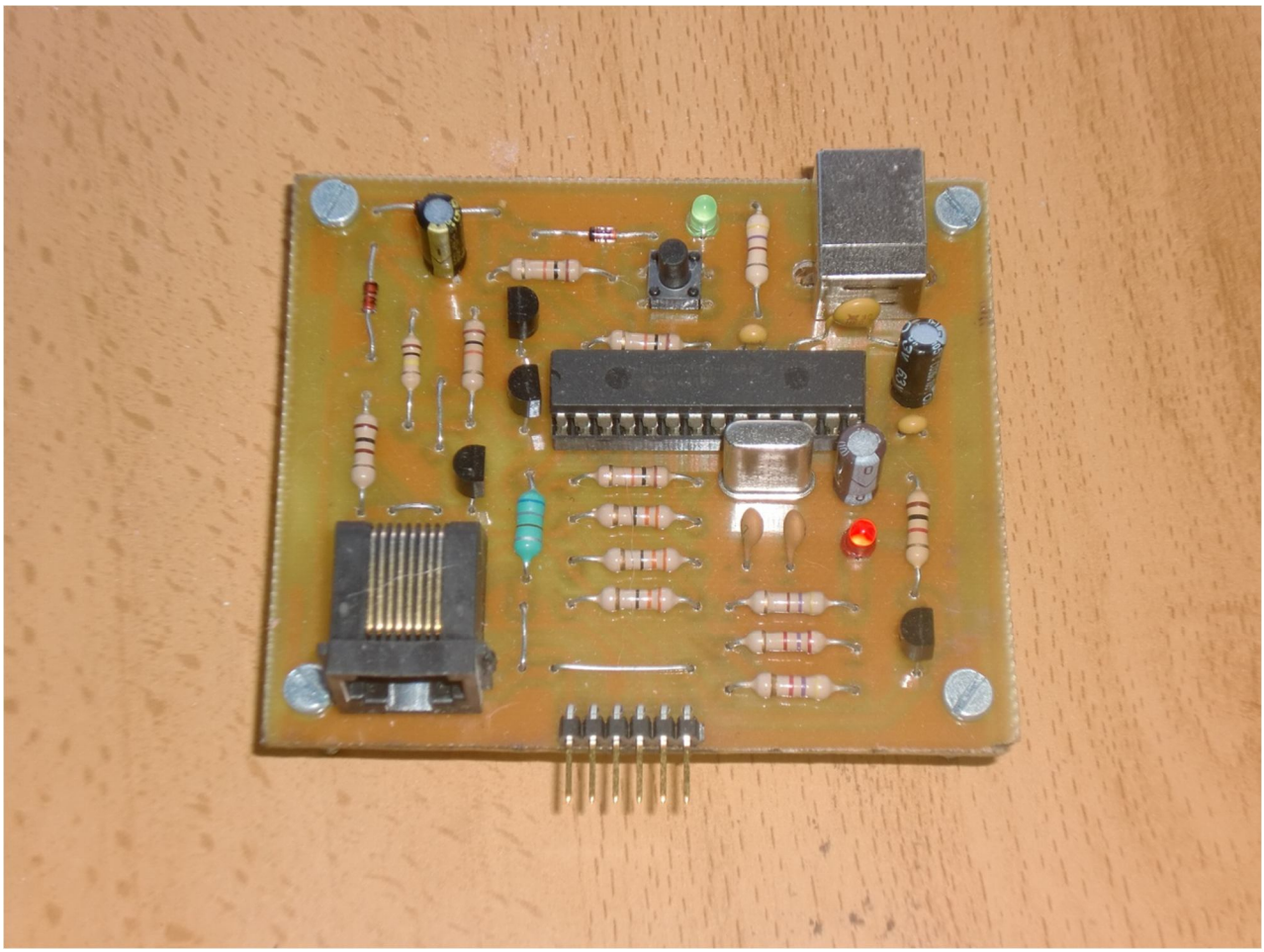
Entrenador y grabador conectados directamente.



Entrenador



Programador



Lista de componentes y materiales utilizados

A continuación se realiza un presupuesto a fecha del 30 de Octubre del 2012 de la página web de Farnell.

El precio, IVA y el total de los presupuestos están expresados en Euros (€).
El precio es el precio unitario. El código es el que utiliza Farnell en su Web.

- Programador PicKit2

Código	Cantidad	Concepto	Precio	Total
9340440	3	Resistencia 33 Ω ½ W	0,043	0,129
9339760	1	Resistencia 100 Ω ½ W	0,042	0,042
1357858	2	Resistencia 470 Ω ½ W	0,250	0,500
9339779	1	Resistencia 1 KΩ ½ W	0,043	0,043
9340319	1	Resistencia 2K7 ½ W	0,043	0,043
1357865	2	Resistencia 4K7 ½ W	0,043	0,086
9339787	4	Resistencia 10 KΩ ½ W	0,043	0,172
9339795	1	Resistencia 100 KΩ ½ W	0,042	0,042
1827806	2	Condensador cerámico 15 pF	0,191	0,382
1216444	2	Condensador cerámico 100 nF	0,179	0,358
1902913	1	Condensador electrolítico 10 µF / 25 V	0,051	0,051
1136296	2	Condensador electrolítico 47 µF / 25 V	0,064	0,128
1180377	1	Bobina 680 µH	0,200	0,200
1467882	1	Transistor BC557	0,119	0,119
1467872	3	Transistor BC548	0,144	0,432
1611492	3	Diodos 1N4148	0,067	0,201
1855559	1	Led color verde	0,360	0,360
1855560	1	Led color rojo	0,290	0,290
9321250	1	Microcontrolador PIC18F2550	5,010	5,010
1103850	1	Zócalo de 28 pines	0,970	0,970
9509674	1	Cristal de cuarzo 20 MHz	1,650	1,650
1555985	1	Micropulsador	0,160	0,160
1843616	1	Fusible rearmable 500 mA	0,440	0,440
1308876	1	Conector USB tipo B para PCB	0,700	0,700
2131147	1	Conector RJ45 para PCB	5,040	5,040
1593438	1	Conector 6 pines acodado hembra	0,990	0,990
1267751	1	Placa PCB fotosensible positiva 100 mm x 160 mm	3,510	3,510
1896047	1	Cable USB tipo A a USB tipo B 1m	7,720	7,720
		Subtotal (€)		**29,768**
		I.V.A. (21%)		**6,251**
		Importe Total (€)		**36,02**

- Entrenador de PICs

Código	Cantidad	Concepto	Precio	Total
1357858	3	Resistencia 470 Ω ½ W	0,250	0,750
9339787	3	Resistencia 10 KΩ ½ W	0,043	0,129
1827806	2	Condensador cerámico 15 pF	0,191	0,382
1216444	3	Condensador cerámico 100 nF	0,179	0,537
1457659	1	Condensador cerámico 330 nF	0,350	0,350
1902913	1	Condensador electrolítico 10 µF / 25 V	0,051	0,051
1136296	2	Condensador electrolítico 47 µF / 25 V	0,064	0,128
1611492	4	Diodo 1N4148	0,067	0,268
1612313	1	Diodo 1N5007 o equivalente (2 A)	0,320	0,320
1855559	2	Led color verde	0,360	0,720
1855560	1	Led color rojo	0,290	0,290
2126406	1	Microcontrolador PIC16F877	4,610	4,610
1077116	1	Interruptor electrónico DG403	3,830	3,830
9756086	1	Regulador L78S05	0,650	0,650
1103855	1	Zócalo de 40 pines	2,530	2,530
1101347	1	Zócalo de 16 pines	0,250	0,250
9509674	1	Cristal de cuarzo 20 MHz	1,650	1,650
1555985	1	Micropulsador	0,160	0,160
1843616	1	Fusible rearmable 500 mA	0,440	0,440
1652206	1	Fusible rearmable 2 A	0,790	0,790
1308876	1	Conector USB tipo B para PCB	0,700	0,700
2131147	1	Conector RJ45 para PCB	5,040	5,040
1368647	1	Conector Vcc Jack para PCB	1,270	1,270
1593438	4	Tira de 10 pines acodados machos	0,990	3,960
1218870	4	Tira de 20 pines rectos hembras	1,330	5,320
1382937	1	Radiador para regulador L78S05	0,790	0,790
1267751	1	Placa PCB fotosensible positiva 100 mm x 160 mm	3,510	3,510
		Subtotal (€)		**39,425**
		I.V.A. (21%)		**8,279**
		Importe Total (€)		**47,70**

Total del Proyecto :

Cantidad	Concepto	Precio	Total
1	Programador para PICs USB (cable USB incluido)		29,768
1	Entrenador para PICs		39,425
	Subtotal (€)		**69,193**
	I.V.A. (21%)		**14,531**
	Importe Total (€)		**83,73**

Software utilizado

Software de Programación y Firmware para el PIC18F2550 :

El firmware utilizado para el PIC18F2550 del programador se encuentra en la página web de Microchip :

http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en023805

Se trata de la versión 2.32 que se encuentra en la sección Downloads. En esa misma sección se encuentra el Software de Programación PicKit2 v.2.61 con soporte para diversos sistemas operativos, nosotros usaremos la versión para Windows.

Cabe destacar que Microchip ha realizado el software de Programación PicKit2 v.2.61 de forma específica para el grabador PicKit2, aunque este grabador también es soportado por el entorno MPLab de Microchip.

a) Firmware para el PIC18F2550 v. 2.32

Este es el firmware del PIC18F2550 que dispone de un programa residente (Bootloader) para comunicarse a través del puerto USB de un PC y poder transferir el código del programa fuente al microcontrolador elegido y así programar los diversos microcontroladores y memorias que se indican en las características del programador PicKit2. En nuestro caso lo utilizaremos para programar el PIC16F877.

Hay que destacar que para introducir el firmware en el primer PIC18F2550 se utilizó el grabador de microcontroladores del laboratorio de SED de la Escuela Politécnica Superior de Gandía, una vez programado el primer PIC, y una vez montado el primer programador, los siguientes pueden ser programador por el anterior.

b) Programa de Grabación PicKit2 v.2.61 de Microchip

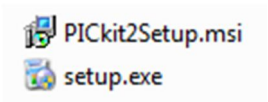
Esta aplicación de programación te permite programar todos los dispositivos soportados en el archivo léeme del PicKit2.

Una vez instalado se puede crear un icono de acceso rápido en el escritorio cuya apariencia es la siguiente :

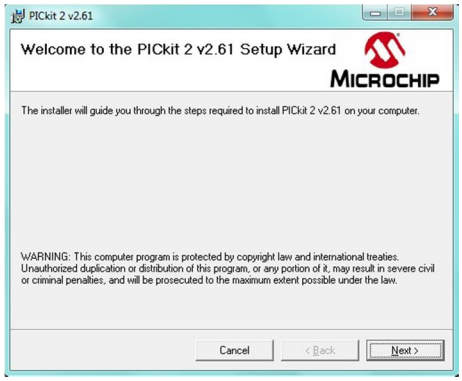
Instalación de la aplicación PicKit 2 Programmer v.2.61

Una vez descargado y descomprimido el fichero de la página de Microchip, obtenemos la siguiente carpeta :

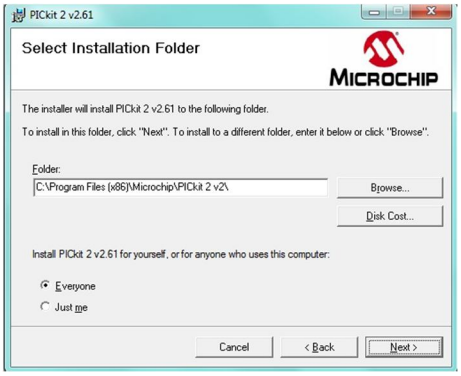
Dentro de esa carpeta tendremos los siguientes ficheros :



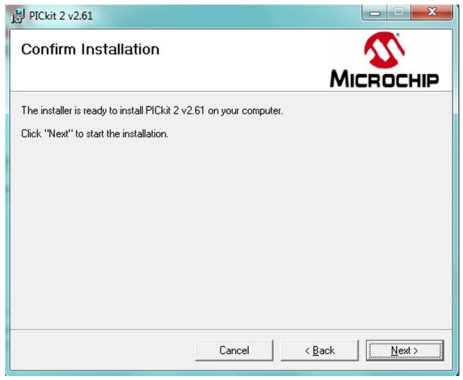
Ejecutamos el fichero setup.exe como Administrador, y seguimos las siguientes instrucciones :



Pulsamos “Next”

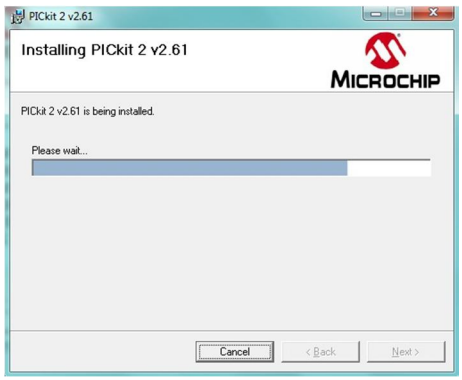


Seleccionamos la carpeta de instalación y pulsamos Next”

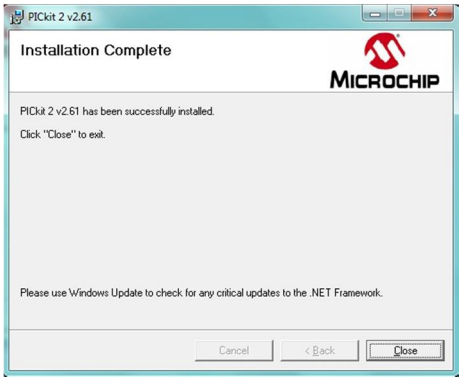


Pulsamos “Next”

Pulsamos “Next”

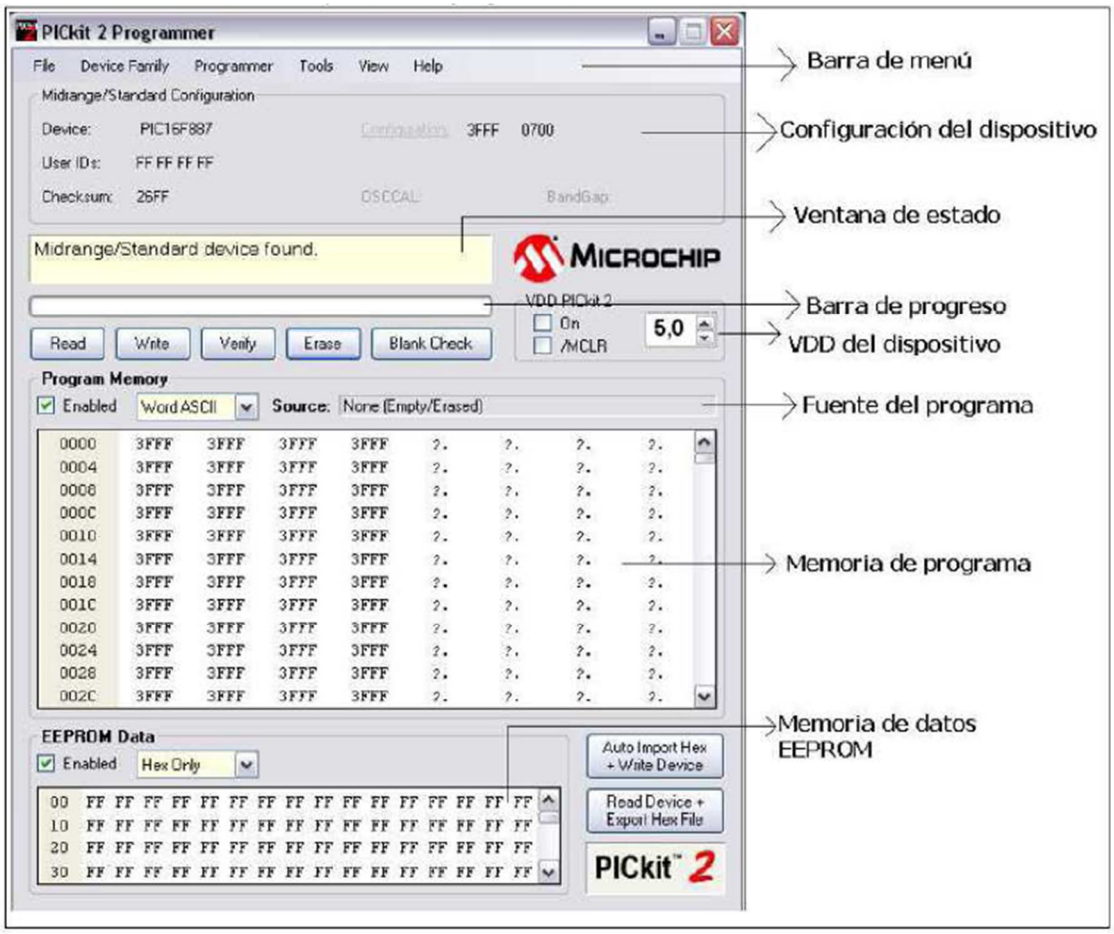


Esperamos a que se instale



Pulsamos “Close”

Una vez finalizada la instalación ejecutaremos la aplicación, tras unos segundos nos aparecerá la siguiente ventana :



La Barra de menú:

En la barra de menú seleccionamos varias de las funciones de la aplicación de programación del PicKit2. Un sumario de esas funciones es:

- **Archivo (File)**
    a) Importar HEX – (Import HEX): Importa un archivo HEX para programar. El formato de archivo HEX, INHX32 está soportado.
    b) Exportar HEX – (Export HEX): Exporta un archivo HEX leído desde el dispositivo. El archivo HEX es creado en el formato INHX32.
    c) Historial de archivo – (File History): Los últimos 4 archivos HEX abiertos son mostrados con la dirección de donde se encuentran. Estos archivos recién abiertos son seleccionados para la rápida importación.
    d) Exit: Sale del programa.

- **Familia de dispositivos (Device Family)**

  Selecciona la familia de dispositivos la cual borrará todos los datos de dicho dispositivo.

- **Programador (Programmer)**
  a) Leer dispositivo (Read device): Lee la memoria de programa, la memoria eeprom de datos, las locaciones ID y los bits de configuración.
  b) Programar dispositivo (Write device): Escribe la memoria de programa, la memoria eeprom de datos, las locaciones ID y los bits de configuración.
  c) Verificar (Verify): Verifica la memoria de programa, la memoria eeprom de datos, locaciones ID y los bits de configuración leídos comparando con los códigos guardados en el programa de aplicación.
  d) Borrar (Erase): Realiza un borrado general del microcontrolador seleccionado.
  e) Chequeo de blanco (Blank check): Realizamos un chequeo para ver si la memoria de programa, la memoria eeprom, los bits de configuración y las locaciones de ID están en blanco (vacio).
  f) Verificación en la escritura (Verify on write): Si seleccionamos esta opción el dispositivo es verificado después de que es programado. En cambio si no seleccionamos la opción, el dispositivo es programado pero no se verifica después de la programación.
  g) Mantener el dispositivo en reset (Hold device in reset): Cuando seleccionamos esta opción el PIN /MCLR, es mantenido a nivel bajo. Cuando no seleccionamos esta opción el pin es liberado (modo triestado), permitiendo que una resistencia externa de pull–up saque el dispositivo del estado de reset.
  h) Escribir (opción) – Botón del Pickit2 (Write on – Pickit2 button): Cuando esta opción es seleccionada, una operación de escritura deberá ser iniciada presionando el botón del pickit2.

- **Herramientas (Tools)**
  a) Habilitar protección de código (Enable code Protect): Habilita las características de protección de código del microcontrolador en futuras operaciones de escritura.

     Nota: Para desactivar la protección de código, toda la memoria debe ser borrada y rescrita.

Para mayor detalle de la herramienta de programación se suministra el Manual completo de Microchip junto con esta memoria.

Software MPLAB X de Microchip :

MPLAB es un editor IDE gratuito, destinado a productos de la marca Microchip. Este editor es modular, permite seleccionar los distintos microcontroladores soportados, además de permitir la grabación de estos circuitos integrados directamente al programador.

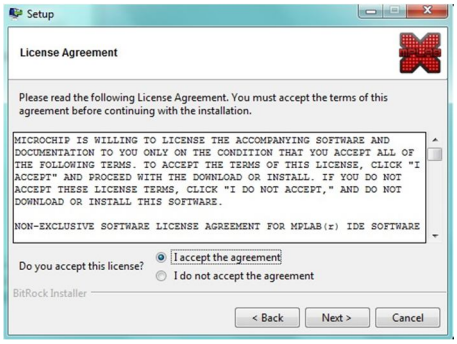
Es un programa que corre bajo Windows y como tal, presenta las clásicas barras de programa, de menú, de herramientas de estado, etc. El ambiente MPLAB® posee editor de texto, compilador y simulación (no en tiempo real).

Podemos descargar el software de forma gratuita de la web de Microchip, y podemos elegir diversos sistemas operativos, nosotros seleccionaremos la plataforma Windows y concretamente para versiones de 64 bits.

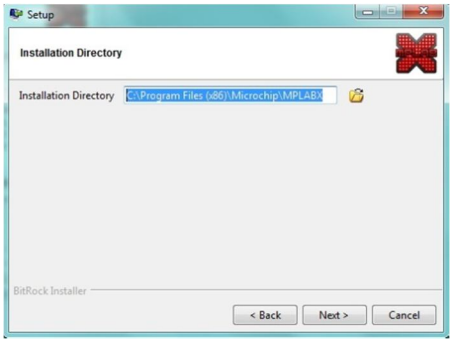
http://www.microchip.com/pagehandler/en-us/family/mplabx/#downloads

Tras lanzar el fichero de instalación obtendremos las siguientes pantallas :

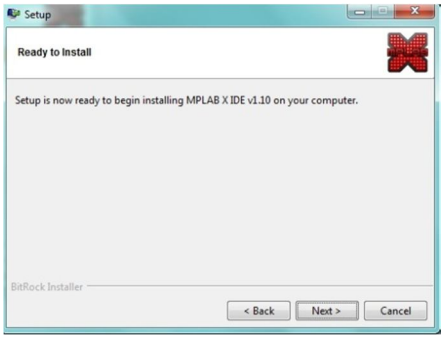
Pulsamos Next



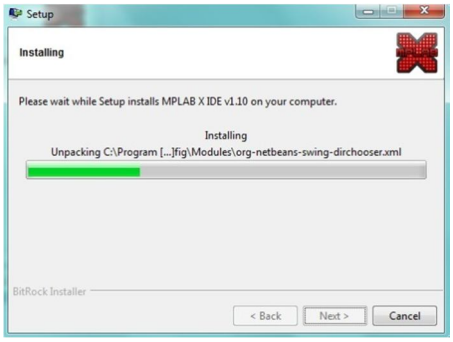
Aceptamos y pulsamos Next



Seleccionamos la carpeta y pulsamos Next

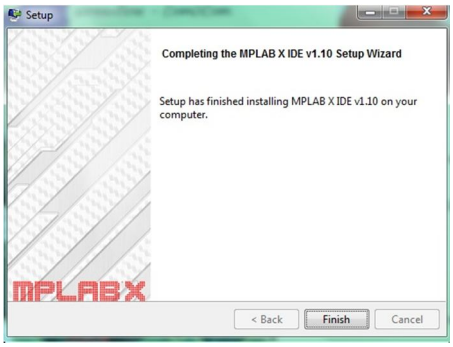


Pulsamos Next



En este caso, "Instalar este Software de controlador de todas formas"

Esperamos a que termine y pulsamos Next



Una vez instalado pulsamos en Finish.

Compilador Hi-Tech C :

El compilador Hi-Tech C es un software que permite programar los microcontroladores PICs en lenguaje ANSI C, puede ser utilizado de forma individual o de forma conjunta con MPLAB de Microchip. La ventaja de utilizar esta herramienta junto con MPLAB reside en el aprovechamiento gratuito del IDE de Microchip y la potencia de este compilador que a pesar de ser de pago, es posible descargarse una versión demo, la cual cubre de forma satisfactoria las necesidades que se exigen en el ámbito de la educación.

Además de este compilador, existen otros 2 que son :

- Compilador CCS : El ambiente de desarrollo integrado de PCWHD le da a los programadores la capacidad de producir rápidamente códigos muy eficientes usando un lenguaje fácil y manejable, pero de alto nivel.
- Microchip PIC18 : Microchip también suministra un compilador de lenguaje C para sus microcontroladores PIC y también es de pago.

Como primera opción y comprobando que los 3 compiladores son de pago, se decidió utilizar el compilador C que Microchip, pero tras realizar una comparación del código generado por los 2 compiladores, se comprobó que el código generado por Microchip era un 137 % más largo que el generado por HI-TECH (http://www.xargs.com/pic/picc18-vs-c18.html). Este hecho fue admitido por Microchip, hasta el punto que Microchip recomienda la utilización de Hi-Tech en su página web

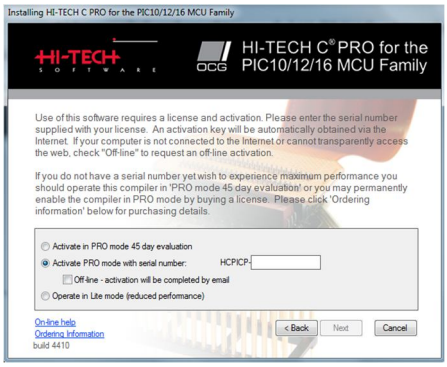
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en542849

Aunque el compilador CCS es utilizado ampliamente también, dado que Microchip recomienda Hi-Tech C, y éste también es utilizado ampliamente, es integrable por MPLAB y además es utilizado en la asignatura de Procesadores avanzados, se decide utilizar este compilador.

Instalación e integración en MPLAB del compilador HI-TECH C :

Al hacer clic sobre el icono del HI-TECH C, obtendremos la siguiente ventana :

Pinchamos en Next



Tendremos que activarlo

Tendremos que decidir, entre comprar el producto e introducir el código de activación que nos suministran, o activar el producto entero para ser utilizado durante un tiempo máximo de 45 días o activar el producto de forma reducida. En el CD que acompaña a este trabajo se adjunta las instrucciones necesarias para activar el producto de forma completa.

Después seguiremos las instrucciones que nos indique, tendremos que indicar la carpeta donde se encuentra instalado el MPLAB.

Al finalizar tendremos instalado y activado el Hitech – C.

Ejercicio práctico : Reloj en tiempo real

Como ejercicio para demostrar el correcto funcionamiento del grabador y del entrenador se ha realizado un reloj en tiempo real, con posibilidad de ponerlo en hora mediante unos switchs externos.

A continuación se adjunta el código empleado para dicho ejercicio.

Realización de un Reloj en tiempo real :

Código main.c

```
/*****************************************************************************
***                              RELOJ EN TIEMPO REAL                      ***
***                    Realizado por : Juan Carlos López Gordillo          ***
***                    E.P.S.G.   Trabajo Final de Carrera 2012 - 2013     ***
*****************************************************************************/

#define _16F877
#define _XTAL_FREQ 20e6
#include "pic.h"
#include "lcd.h"
__CONFIG(0x3F3A);

void mostrarHora(unsigned char opcion);

unsigned char milisegundos=0,segundos=0, minutos=0, horas=0;
unsigned char bHora, bMinutos, bSegundos;
void main (void)
{
  unsigned char auxHora=0, auxMinutos=0, auxSegundos=0;
  TRISD=0xff;
  LCD_Init(); LCD_Cursor("off");
  LCD_Goto(1,1); LCD_Texto("Hora  Min.  Seg.");
  LCD_Goto(1,2); LCD_Texto(" 00  : 00 :  00 ");

  INTCON=0xD0;
  T1CON=0x30; //PRESCALER 1:8
  TMR1H=0x0B; TMR1L=0xDC;   //TMR1H,L = 3036  = 0x0BDC  100 ms
  TMR1IE=1; TMR1ON=1;
  while(1)
  {
    bHora=RD0; bMinutos=RD1; bSegundos=RD2;

    if (bHora==1 && auxHora==0) { auxHora=1; horas++; mostrarHora('h'); }
    if (bHora==0 && auxHora==1) { auxHora=0; }
    if (bMinutos==1 && auxMinutos==0) { auxMinutos=1; minutos++; mostrarHora('m'); }
    if (bMinutos==0 && auxMinutos==1) { auxMinutos=0; }
    if (bSegundos==1 && auxSegundos==0) { auxSegundos=1; segundos=0; mostrarHora('s'); }
    if (bSegundos==0 && auxSegundos==1) { auxSegundos=0; }
  }
}
```

```
void mostrarHora(unsigned char opcion)
{
   char chComun[2];
   char decenas, unidades, comun, pos;

   if (opcion=='s' || opcion=='S')
   {
      if (segundos>=60) { segundos=0; }
      comun=segundos; pos=14;
   }
   if (opcion=='m' || opcion=='M')
   {
      if (minutos>=60) { minutos=0; }
      comun=minutos; pos=8;
   }
   if (opcion=='h' || opcion=='H')
   {
      if (horas>=24) { horas=0; }
      comun=horas; pos=2;
   }

   decenas=comun/10; unidades=comun-(decenas*10);
   chComun[0]=decenas+48; chComun[1]=unidades+48; chComun[2]='\0';
   LCD_Goto(pos,2); LCD_Texto(chComun);
}

void interrupt temporizador(void)
{
   if (TMR1IF==1) // Se produce cada 100 ms
   {
      TMR1H=0x0B; TMR1L=0xDC;   //TMR1H,L = 3036  = 0x0BDC 100ms
      milisegundos++;
      if (milisegundos==5) //cada 500 ms
      {
         LCD_Goto(6,2); LCD_Caracter(' '); LCD_Goto(11,2); LCD_Caracter(' ');
      }
      if (milisegundos==10) // cada segundo
      {
         LCD_Goto(6,2); LCD_Caracter(':'); LCD_Goto(11,2); LCD_Caracter(':');

         milisegundos=0; segundos++;
         if (segundos==60)
         {
            segundos=0; minutos++;
            if (minutos==60)
            {
               minutos=0; horas++;
               if (horas==24) { horas=0; }
               mostrarHora('h');
            }
            mostrarHora('m');
         }
         mostrarHora('s');
      }
      TMR1IF=0;
   }
}
```

Código LCD.h

```
/*************************************************************************
***                    FUNCIONES PARA EL MANEJO DE UN DISPLAY LCD      ***
***                    Realizado por : Juan Carlos López Gordillo      ***
***                    E.P.S.G.   Trabajo Final de Carrera 2012 - 2013 ***
*************************************************************************/
#include "delay.h"
#include <string.h>
#define     LCD_Strobe()            ((EN = 1),(EN=0))

/*************************************************************************
***                    Modificar en función del Puerto que se vaya a utilizar ***
*************************************************************************/
#define PUERTO PORTC     // LCD conectado D4-R4, D5-R5, D6-R6 y D7-R7
#define TRISX  TRISC
#define RS RC0
#define RW RC1
#define EN RC2
//************************************************************************

/*************************************************************************
***                  FUNCIONES                                         ***
*************************************************************************/
extern void LCD_Init(void); // Inicialización del LCD
extern void LCD_Write(unsigned char dato);  // Escribir un byte en LCD
// Borra LCD entero (linea=0) o únicamente la linea indicada
extern void LCD_Clear(unsigned char linea);
extern void LCD_Texto(const char *dato); // Escribit Texto en LCD
// Desplazamiento de Texto en la linea y tiempo en ms indicados
extern void LCD_RotarIzquierda(unsigned char linea, char *texto, int tiempo);
extern void LCD_RotarDerecha(unsigned char linea, char *texto, int tiempo);
// Posicionan el cursor en la linea y posicion (eje x) indicados
extern void LCD_Goto(unsigned char posicion, unsigned char linea);
extern void LCD_Caracter(unsigned char dato); // Escribe un Caracter en LCD
// Centra el texto en la linea indicada
extern void LCD_Centrar(unsigned char linea, const char *texto);
// Definen visibilidad y parpadeo del cursos ("ON", "OFF", "1", "0")
extern void LCD_Cursor(const char *estado); // Define visibilidad del cursor
extern void LCD_Blink(const char *estado); // Define parpadeo del cursor
// Generación de caracteres especiales definidos en la RAM de la LCD
extern void LCD_enye(unsigned char direccion);  // Letra ñ
extern void LCD_acute(unsigned char direccion); // Letra á
extern void LCD_ecute(unsigned char direccion); // Letra é
extern void LCD_icute(unsigned char direccion); // Letra í
extern void LCD_ocute(unsigned char direccion); // Letra ó
extern void LCD_ucute(unsigned char direccion); // Letra ú
//************************************************************************

unsigned char DisplayControl=0x0C;

void LCD_Write(unsigned char dato)
{
   DelayUs(40);
   PUERTO = (dato & 0xF0) | (PUERTO & 0x0F) ; LCD_Strobe();
   PUERTO = ((dato << 4) & 0xF0) | (PUERTO & 0x0F); LCD_Strobe();
}
```

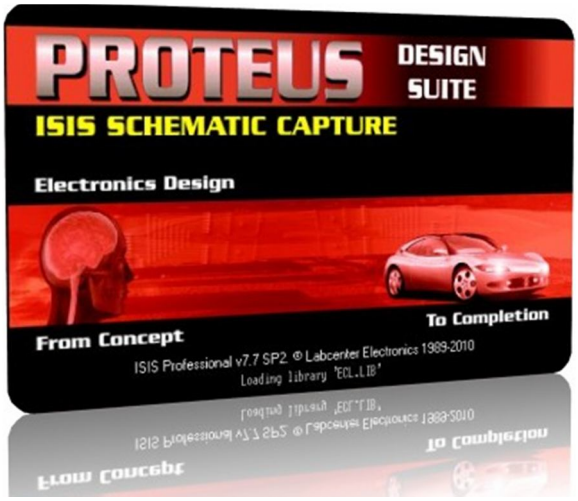
```
void LCD_Clear(unsigned char linea)
{
   if (linea>2) { linea=0; }
   if (linea==0)
   {
      RS=0;
      LCD_Write(0x01); DelayMs(2);
   }
   if (linea==1 || linea==2)
   {
      RS=1;
      LCD_Goto(1,linea); LCD_Texto("               "); LCD_Goto(1,linea);
   }
}

void LCD_Texto(const char *dato)
{
   RS=1;
   while(*dato) { LCD_Write(*dato++); }
}

void LCD_RotarIzquierda(unsigned char linea, char *texto, int tiempo)
{
   unsigned char x,x1;
   unsigned int longitud;
   char frase[32], frase2[32];

   strcpy(frase,texto);
   longitud=strlen(frase);
   LCD_Cursor("Off");
   LCD_Clear(linea);
   for (x=16;x>=1; x--)
   {
      LCD_Goto(x,linea); LCD_Texto(frase);
      LCD_Goto(x+longitud,linea); LCD_Caracter(' ');
      DelayMs(tiempo);
   }
   for (x=1; x<=longitud-1; x++)
   {
      for(x1=0;x1<=longitud-1-x;x1++)
      {
         frase2[x1]=frase[x+x1];
      }
      frase2[x1]='\0';
      LCD_Goto(1,linea); LCD_Texto(frase2);
      LCD_Goto(longitud-x+1,linea); LCD_Caracter(' ');
      DelayMs(tiempo);
   }
   LCD_Goto(1,linea); LCD_Caracter(' ');
}
void LCD_RotarDerecha(unsigned char linea, char *texto, int tiempo)
{
   unsigned char x,x1;
   unsigned int longitud;
   char frase[32], frase2[32];

   strcpy(frase,texto);
   longitud=strlen(frase);
   LCD_Cursor("Off");
   LCD_Clear(linea);

   for (x1=longitud-1; x1>=1; x1--)
   {
      for (x=0; x<=longitud-1-x1; x++)
      {
         frase2[x]=frase[x1+x];
      }
      frase2[x]='\0';
      LCD_Goto(1,linea); LCD_Texto(frase2);
      DelayMs(tiempo);
   }
```

```
   for (x=2; x<=17; x++)
   {
      LCD_Goto(x,linea); LCD_Texto(frase);
      LCD_Goto(x-1,linea); LCD_Caracter(' ');
      DelayMs(tiempo);
   }
}

void LCD_Caracter(unsigned char dato)
{
   RS=1;
   LCD_Write(dato);
}

void LCD_Goto(unsigned char posicion, unsigned char linea)
{
   posicion--;
   if (linea<1 || linea>2) { linea=1; }
   RS=0;
   if (linea==1){ LCD_Write(0x80+posicion); }
   if (linea==2){ LCD_Write(0x80+0x40+posicion); }
}
void LCD_Centrar(unsigned char linea, const char *texto)
{
   unsigned int longitud;
   unsigned char x;

   x=1; longitud=strlen(texto);
   if (longitud<15) { x=(8-(longitud/2))+1; }
   LCD_Goto(x,linea); LCD_Texto(texto);
}

void LCD_Cursor(const char *estado)
{
   if (estado=="ON" || estado=="On" || estado=="on" || estado=="1")
   { DisplayControl=DisplayControl | 0x02; }
   else
   { DisplayControl=DisplayControl & 0xFD; }
   LCD_Write(DisplayControl);
}

void LCD_Blink(const char *estado)
{
   if (estado=="ON" || estado=="On" || estado=="on" || estado=="1")
   { DisplayControl=DisplayControl | 0x01; }
   else
   { DisplayControl=DisplayControl & 0xFE; }
   LCD_Write(DisplayControl);
}
void LCD_enye(unsigned char direccion)
{
   RS=0;
   LCD_Write(0x40+(direccion*8)); DelayUs(40);
   RS=1;
   LCD_Write(0x0E); DelayUs(40);
   LCD_Write(0x00); DelayUs(40);
   LCD_Write(0x16); DelayUs(40);
   LCD_Write(0x19); DelayUs(40);
   LCD_Write(0x11); DelayUs(40);
   LCD_Write(0x11); DelayUs(40);
   LCD_Write(0x11); DelayUs(40);
   LCD_Write(0x00); DelayUs(40);
   LCD_Goto(1,1);
}
```

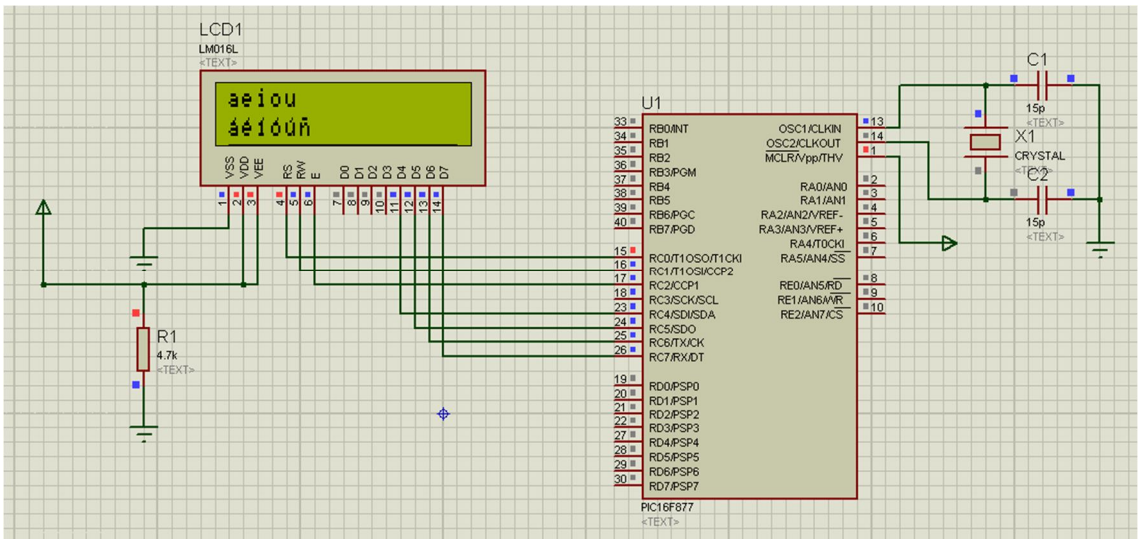
```
void LCD_acute(unsigned char direccion)
{
   RS=0;
   LCD_Write(0x40+(direccion*8)); DelayUs(40);
   RS=1;
   LCD_Write(0x02); DelayUs(40);
   LCD_Write(0x04); DelayUs(40);
   LCD_Write(0x0E); DelayUs(40);
   LCD_Write(0x01); DelayUs(40);
   LCD_Write(0x0F); DelayUs(40);
   LCD_Write(0x11); DelayUs(40);
   LCD_Write(0x0F); DelayUs(40);
   LCD_Write(0x00); DelayUs(40);
   LCD_Goto(1,1);
}
void LCD_ecute(unsigned char direccion)
{
   RS=0;
   LCD_Write(0x40+(direccion*8)); DelayUs(40);
   RS=1;
   LCD_Write(0x02); DelayUs(40);
   LCD_Write(0x04); DelayUs(40);
   LCD_Write(0x0E); DelayUs(40);
   LCD_Write(0x11); DelayUs(40);
   LCD_Write(0x1F); DelayUs(40);
   LCD_Write(0x10); DelayUs(40);
   LCD_Write(0x0E); DelayUs(40);
   LCD_Write(0x00); DelayUs(40);
   LCD_Goto(1,1);
}
void LCD_icute(unsigned char direccion)
{
   RS=0;
   LCD_Write(0x40+(direccion*8)); DelayUs(40);
   RS=1;
   LCD_Write(0x02); DelayUs(40);
   LCD_Write(0x04); DelayUs(40);
   LCD_Write(0x0C); DelayUs(40);
   LCD_Write(0x04); DelayUs(40);
   LCD_Write(0x04); DelayUs(40);
   LCD_Write(0x04); DelayUs(40);
   LCD_Write(0x0E); DelayUs(40);
   LCD_Write(0x00); DelayUs(40);
   LCD_Goto(1,1);
}
void LCD_ocute(unsigned char direccion)
{
   RS=0;
   LCD_Write(0x40+(direccion*8)); DelayUs(40);
   RS=1;
   LCD_Write(0x02); DelayUs(40);
   LCD_Write(0x04); DelayUs(40);
   LCD_Write(0x0E); DelayUs(40);
   LCD_Write(0x11); DelayUs(40);
   LCD_Write(0x11); DelayUs(40);
   LCD_Write(0x11); DelayUs(40);
   LCD_Write(0x0E); DelayUs(40);
   LCD_Write(0x00); DelayUs(40);
   LCD_Goto(1,1);
}
```

```
void LCD_ucute(unsigned char direccion)
{
   RS=0;
   LCD_Write(0x40+(direccion*8)); DelayUs(40);
   RS=1;
   LCD_Write(0x02); DelayUs(40);
   LCD_Write(0x04); DelayUs(40);
   LCD_Write(0x11); DelayUs(40);
   LCD_Write(0x11); DelayUs(40);
   LCD_Write(0x11); DelayUs(40);
   LCD_Write(0x13); DelayUs(40);
   LCD_Write(0x0D); DelayUs(40);
   LCD_Write(0x00); DelayUs(40);
   LCD_Goto(1,1);
}

void LCD_Init(void)
{
   TRISX=0x00; RS=0; EN=0; RW=0; DelayMs(15);
   PUERTO = ((0x03<<4) & 0xF0) | (PUERTO & 0x0F);
   LCD_Strobe(); DelayMs(5);
   LCD_Strobe(); DelayUs(200);
   LCD_Strobe(); DelayUs(200);
   PUERTO = 0x20 | (PUERTO & 0x0F); // modo de 4 bits
   LCD_Strobe();
   LCD_Write(0x28); // modo 4 bits, 2 lineas, 5x7 dots
   LCD_Write(DisplayControl); //Display ON
   LCD_Clear(0); // borro pantalla
   LCD_Write(0x06); // habilito para escribir
   /*************************************************************************
   **          Letra ñ : Dirección 0  Letra á : Dirección 1  Letra é : Dirección 2          **
   **          Letra í : Dirección 3  Letra ó : Dirección 4  Letra ú : Dirección 5          **
   *************************************************************************/
   LCD_enye(0); LCD_acute(1); LCD_ecute(2);
   LCD_icute(3); LCD_ocute(4); LCD_ucute(5);
}
```

Para asegurarse de que tanto el hardware como el software funciona correctamente se ha utilizado el Software Proteus v.7.7 Sp2, el cual dispone de la posibilidad de simular una amplia gama de microcontroladores, entre ellos los microcontroladores PIC usados en este trabajo.



En primer lugar se realizó un programa para controlar el display LCD y realizar diversas funciones para su posterior utilización. La simulación realizada con Proteus ha sido la siguiente :



Como se observa, además de probar diversas funciones básicas, también se ha implementado la posibilidad de añadir a nuestros proyectos las vocales acentuadas y la letra ñ.

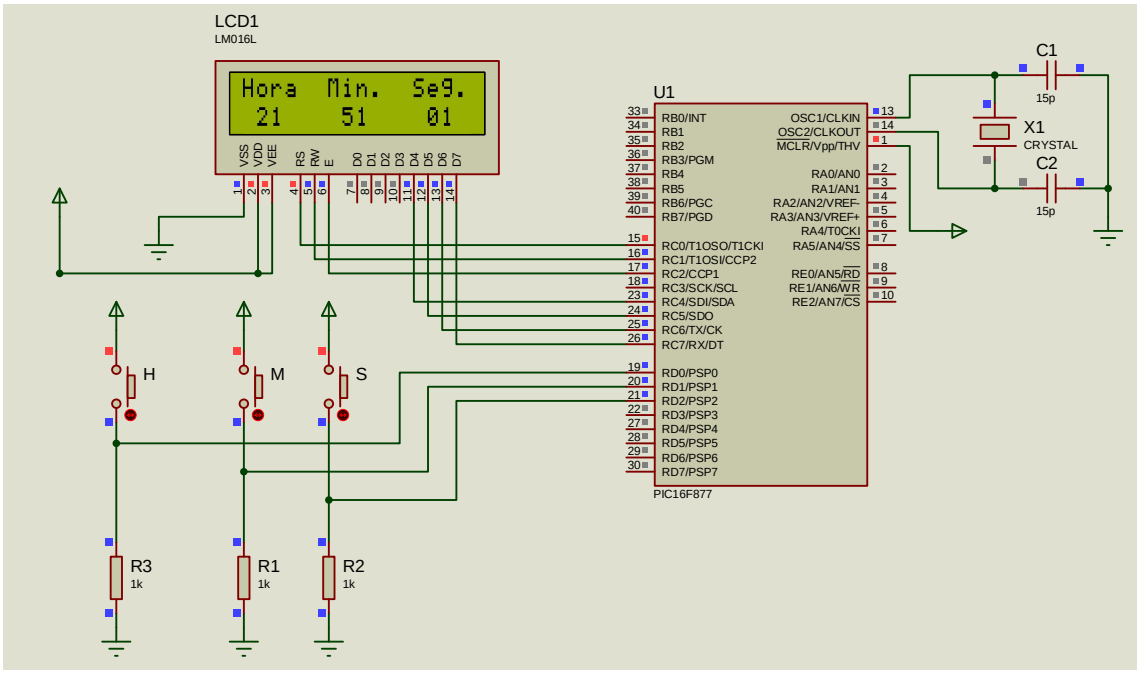
El programa de prueba se suministra también el CD adjunto y se expone a continuación :

```
#define _16F877
#define _XTAL_FREQ 20e6
#include "pic.h"
#include "lcd.h"
__CONFIG(0x3F3A);

void main (void)
{
        LCD_Init();
        LCD_Cursor("off");
        //LCD_Goto(1,1); LCD_Texto("Espa");LCD_Caracter(0);LCD_Texto("a");
        //LCD_Goto(1,2); LCD_Texto("HOLA MUNDO!!!");
        LCD_Goto(1,1); LCD_Texto("aeiou");
        LCD_Goto(1,2); LCD_Caracter(1);
        LCD_Goto(2,2); LCD_Caracter(2);
        LCD_Goto(3,2); LCD_Caracter(3);
        LCD_Goto(4,2); LCD_Caracter(4);
        LCD_Goto(5,2); LCD_Caracter(5);
        LCD_Goto(6,2); LCD_Caracter(0);
        //LCD_Centrar(2,"H2");
        //LCD_Clear(2);
        //LCD_RotarIzquierda(1,"Prueba de izquierda larga",1000);
        //LCD_RotarDerecha(1,"Esto es una prueba derecha larga",1000);
        while(1);
}
```

Debemos fijarnos que se trata de un programa que sirve para probar las funciones que se han ido desarrollando para el manejo del display LCD, de ahí que existan líneas de código en comentarios.

También debo señalar que las funciones están implementadas en el fichero lcd.h incluido al principio del código.

La simulación del proyecto completo con Proteus también se incluye en el CD adjunto y es la siguiente :



Como se observa, se dispone de 3 pulsadores para poner en hora el reloj, permite ajustar las horas, ajustar los minutos y resetear los segundos. El programa utilizado ya se ha indicado en páginas anteriores.

Conclusiones

Los objetivos planteados al iniciar el presente trabajo final de carrera eran :

- Estudio y montaje de un programador y depurador para PICs compatible con el software de Microchip.
- Estudio y depuración de software y hardware utilizando el puerto USB.
- Realización de un entrenador de PICs usando el puerto USB.
- Adquirir un nivel de especialización mayor en la programación y manejo de los PICs.
- Estudio del modo de programación ICSP utilizado en los microcontroladores actuales.
- Realizar un sistema conjunto de Programador y Entrenador compatible con Microchip para ser utilizado en equipos actuales que utilicen Windows 7.

Se puede observar, que se han cumplido todos los objetivos planteados, además se han adquirido otros conocimientos no programados en los objetivos iniciales, tales como la utilización de un software de simulación de microcontroladores (Proteus).

Como conclusión final tengo que exponer que este trabajo final de carrera me ha servido para adquirir unos conocimientos más amplios acerca del funcionamiento del puerto USB, así como de los microcontroladores PIC.

La realización del proyecto desde su inicio ha sido bastante laboriosa, ya que he tenido que recopilar información del funcionamiento del programador PicKit2 original de microchip, estudiar tanto el hardware como el software que Microchip proporciona y adaptarlo para poder realizarlo con componentes electrónicos de fácil adquisición.

Una vez realizado los esquemas iniciales, con sus respectivas simulaciones para comprobar su correcto funcionamiento, la siguiente fase, fue la realización del PCB, el cual ha sufrido bastantes modificaciones para optimizar el funcionamiento y tamaño dela placa de circuito impreso.

Tras elaborar el PCB definitivo, y comprobar que funcionaba según lo previsto, se elaboró el software tanto para la programación como para comprobar el funcionamiento correcto de todo el sistema diseñado.

Por último, todo lo realizado anteriormente se ha plasmado en la presente memoria.

Futuras líneas del trabajo

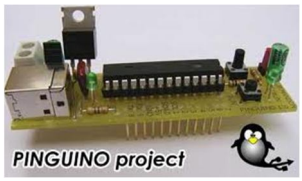
Una continuación del presente trabajo, podría ser la elaboración de otros entrenadores para los PICs que Microchip está desarrollando tales como los dspPICs, y la posible mejora de realizar un proyecto similar al actual, pero teniendo como referencia el PicKit 3 de Microchip, el cual dispone de mayores prestaciones que el PicKit 2.

Debe tenerse en cuenta que los microcontroladores se encuentran en constante evolución y las herramientas con las que se programan (básicamente PCs) también, por lo que el presente trabajo se podría mejorar, realizando un programador que utilice los Puertos USB 3.0, compatibles con Windows 8, etc…

Podría mencionar infinidad de líneas de trabajo usando microcontroladores PIC, pero hay una que creo que merece especial mención por el auge que está teniendo, me refiero a la placa Arduino, que utiliza un microcontrolador Atmel y que debido a sus numerosos shields (accesorios) que existen y a la facilidad de diseñar de una manera rápida diferentes diseños, está teniendo mucho auge en la educación. Además también se está utilizando en el campo de la Domótica libre (OpenDomo).

http://es.opendomo.org/

Por estos motivos, un futura línea de trabajo, bastante interesante sería realizar un sistema compatible con Arduino, que utilice el mismo software que Arduino, pero con microcontroladores PICs, de esta forma se podría utilizar todos los shields de Arduino con PICs.



Tengo que mencionar que ya existe un proyecto que realiza la función de la placa de Arduino con microcontroladores PIC, se llama Pingüino y se puede consultar en la siguiente web :

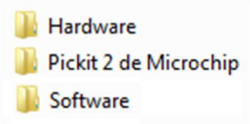
http://www.pinguino.org.ve/~pinguino/wiki/index.php?title=Introducci%C3%B3n_General_a_Ping%C3%BCino

Contenido del CD

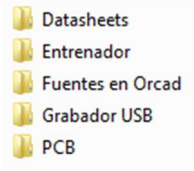
A continuación se expone el contenido del CD adjunto de este trabajo final de carrera :

En la carpeta raíz del CD se encuentran 2 archivos, uno en formato Word 2007 (Memoria.docx) y otro en formato Adobe Acrobat Reader (Memoria.pdf), que corresponden a la memoria del trabajo final de carrera.

También se adjunta las siguientes carpetas :



La carpeta Hardware contiene todo lo relacionado con el trabajo final de carrera en cuanto al hardware se refiere, es decir contiene :



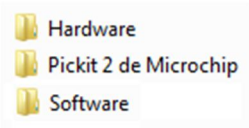
En Datasheets se encuentran todos los datasheets de los fabricantes de los componentes utilizados en el grabador y en el entrenador diseñado.

En la carpeta Entrenador, se encuentran los ficheros creados en Orcad del esquemático , en formato pdf para su impresión en caso de no disponer del paquete Orcad.

En la carpeta Grabador USB, se encuentran los ficheros creados en Orcad del esquemático, en formato pdf para su impresión en caso de no disponer del paquete Orcad.

En la carpeta PCB, se encuentran diversos ficheros en formato pdf de la placa de circuito impreso (cara de pistas, cara de componentes, taladros, etc...).

En la carpeta Fuentes en Orcad, se encuentran los ficheros creados en Orcad del esquemático y del PCB del trabajo final de carrera diseñado.



Volviendo otra vez a la carpeta raíz del CD podemos observar que en la carpeta Pickit 2 de Microchip, se encuentra toda la documentación relacionada con el grabador original que Microchip comercializa.

En la carpeta Software, se ha introducido todo el software que se ha utilizado en este trabajo, tanto el que microchip suministra en su web como de los programas creados por el Autor de este trabajo.

La carpeta Software tiene las siguientes carpetas :

La carpeta Código Fuente contiene los códigos fuente de los programas que Microchip suministra, estos códigos fuentes han servido para que el grabador diseñado sea compatible con el grabador original de Microchip.

La carpeta Compiladores, contiene los compiladores de Hitech-C y C18 que aunque en el trabajo se ha utilizado Hitech-C, se ha decidido adjuntar el C18 por si se desea utilizar este compilador en lugar del Hitech-C.

La carpeta FirmwareV2-32-00 contiene el fichero que Microchip suministra para grabar el PIC18F2550.

La carpeta MPLAB contiene el software MPLAB de Microchip necesario para trabajar con los microcontroladores PIC.

La carpeta PICkit 2 v2.61.00 Setup A, contiene el software de grabación que Microchip suministra junto con el grabador PicKit 2 original de forma gratuita. Hay que destacar que el grabador diseñado es totalmente compatible con el original de Microchip, por lo que puede ser utilizado con total normalidad.

En la carpeta Ejercicio Práctico se encuentra, el ejercicio realizado como demostración del funcionamiento del grabador y del entrenador diseñado. Dentro de esta carpeta se encuentra también los ficheros de simulación creados por el Proteus.

Trabajo realizado por :

Juan Carlos López Gordillo
I.T. Telecomunicaciones
Sistemas Electrónicos

Bibliografía

- Power Electronics. Converters, Applications, and Design (2ª edición).
  N. Mohan, T. M. Undeland, W. P. Robbins, Editorial: John Wiley & Sons, 1995.

- Electrónica Industrial: Técnicas de Potencia (2ª edición).
  Editorial: Marcombo-Boixareu Editores, 1992

- Electrónica de Potencia - Circuitos, Dispositivos y Aplicaciones.
  Muhammad H. Rashid, Prentice Hall Hispanoamericana, S.A., 1993.

- Microcontroladores PICs. 1ª Parte.
  Editorial : McGraw Hill
  José Mª Angulo Usategui

- Microcontroladores PICs 16F87x. 2ª Parte. Diseño Práctico de Aplicaciones.
  Editorial : McGraw Hill
  José Mª Angulo Usategui

- USB Complete
  The Developer's Guide, Fourth Edition
  Jan Axelson

- Proteus. Simulación de circuitos electrónicos y microcontroladores a través de ejemplos.
  Editorial : Marcombo
  Germán Tojeiro Calaza

Links utilizados

- http://www.microchip.com/
- http://www.usb.org
- http://www.htsoft.com/
- http://www.xargs.com/pic/picc18-vs-c18.html
- http://es.farnell.com/

ANEXO

MICROCHIP

PICKIT™ 2
PROGRAMMER-TO-GO
USER GUIDE